Fingercasting–Joint Fingerprinting and Decryption of Broadcast Messages*

André Adelsbach, Ulrich Huber, and Ahmad-Reza Sadeghi

Horst Görtz Institute for IT Security
Ruhr-Universität Bochum
Universitätsstraße 150
D-44780 Bochum
Germany
andre.adelsbach@nds.rub.de, {huber,sadeghi}@crypto.rub.de

Abstract. We propose a stream cipher that provides confidentiality, traceability and renewability in the context of broadcast encryption assuming that collusion-resistant watermarks exist. We prove it to be as secure as the generic pseudo-random sequence on which it operates. This encryption approach, termed fingercasting, achieves joint decryption and fingerprinting of broadcast messages in such a way that an adversary cannot separate both operations or prevent them from happening simultaneously. The scheme is a combination of a known broadcast encryption scheme, a well-known class of fingerprinting schemes and an encryption scheme inspired by the Chameleon cipher. It is the first to provide a formal security proof and a non-constant lower bound for resistance against collusion of malicious users, i.e., a minimum number of content copies needed to remove all fingerprints. To achieve traceability, the scheme fingerprints the receivers' key tables such that they embed a fingerprint into the content during decryption. The scheme is efficient and includes parameters that allow, for example, to trade-off storage size for computation cost at the receiving end.

Keywords: Chameleon encryption, stream cipher, spread-spectrum watermarking, fingerprinting, collusion resistance, frame-proofness, broadcast encryption.

1 Introduction

Experience shows that adversaries attack Broadcast Encryption (BE) systems in a variety of different ways. Their attacks may be on the hardware that stores cryptographic keys, e.g., when they extract keys from a compliant device to develop a pirate device such as the DeCSS software that circumvents the Content Scrambling System [2]. Alternatively, their attacks may be on the decrypted content, e.g., when a legitimate user shares decrypted content with illegitimate users on a file sharing system such as Napster, Kazaa, and BitTorrent.

* An extended abstract of this paper appeared in the Proceedings of the Tenth Australasian Conference on Information Security and Privacy (ACISP 2006) [1].

Y.Q. Shi (Eds.): Transactions on DHMS II, LNCS 4499, pp. 1–34, 2007.

The broadcasting sender thus has three security requirements: *confidentiality*, *traceability* of content and keys, and *renewability* of the encryption scheme. The requirements cover two aspects. Confidentiality tries to prevent illegal copies in the first place, whereas traceability is a second line of defense aimed at finding the origin of an illegal copy (content or key). The need for traceability originates from the fact that confidentiality may be compromised in rare cases, e.g., when a few users illegally distribute their secret keys. Renewability ensures that after such rare events, the encryption system can recover from the security breach.

In broadcasting systems deployed today, e.g., Content Protection for Pre-Recorded Media [3] or the Advanced Access Content System [4], confidentiality and renewability often rely on BE because it provides short ciphertexts while at the same time having realistic storage requirements in devices and acceptable computational overhead. Traitor tracing enables traceability of keys, whereas fingerprinting provides traceability of content. Finally, renewability may be achieved using revocation of the leaked keys.

However, none of the mentioned cryptographic schemes covers all three security requirements. Some existing BE schemes lack traceability of keys, whereas no practically relevant scheme provides traceability of content [5,6,7,8]. Traitor tracing only provides traceability of keys, but not of content [9,10]. Fingerprinting schemes alone do not provide confidentiality [11]. The original Chameleon cipher provides confidentiality, traceability and a hint on renewability, but with a small constant bound for collusion resistance and, most importantly, without formal proof of security [12]. Asymmetric schemes, which provide each compliant device with a certificate and accompany content with Certificate Revocation Lists (CRLs), lack traceability of content and may reach the limits of renewability when CRLs become too large to be processed by real-world devices. Finally, a trivial combination of fingerprinting and encryption leads to an unacceptable transmission overhead because the broadcasting sender needs to sequentially transmit each fingerprinted copy.

Our Contribution. We present, to the best of our knowledge, the first rigorous security proof of Chameleon ciphers, thus providing a sound foundation for the recent applications of these ciphers, e.g., [13]. Furthermore, we give an explicit criterion to judge the security of the Chameleon cipher's key table. Our *fingercasting* approach fulfills all three security requirements at the same time. It is a combination of (i) a new Chameleon cipher based on the *finger*printing capabilities of a well-known class of watermarking schemes and (ii) an arbitrary broad*cast* encryption scheme, which explains the name of the approach. The basic idea is to use the Chameleon cipher for combining decryption and fingerprinting. To achieve renewability, we use a BE scheme to provide fresh session keys as input to the Chameleon scheme. To achieve traceability, we fingerprint the receivers' key tables such that they embed a fingerprint into the content during decryption. To enable higher collusion resistance than the original Chameleon scheme, we tailor our scheme to emulate any watermarking scheme whose coefficients follow a

probability distribution that can be disaggregated into additive components.[1] As proof of concept, we instantiate the watermarking scheme with Spread Spectrum Watermarking (SSW), which has proven collusion resistance [14,15]. However, we might as well instantiate it with any other such scheme.

Joint decryption and fingerprinting has significant advantages compared to existing methods such as transmitter-side or receiver-side Fingerprint Embedding (FE) [11]. Transmitter-side FE is the trivial combination of fingerprinting and encryption by the sender. As discussed above, the transmission overhead is in the order of the number of copies to be distributed, which is prohibitive in practical applications. Receiver-side FE happens in the user's receiver; after distribution of a single encrypted copy of the content, a secure receiver based on tamper-resistant hardware is trusted to embed the fingerprint *after* decryption. This saves bandwidth on the broadcast channel. However, perfect tamper-resistance cannot be achieved under realistic assumptions [16]. An adversary may succeed in extracting the keys of a receiver and subsequently decrypt without embedding a fingerprint.

Our fingercasting approach combines the advantages of both methods. It saves bandwidth by broadcasting a single encrypted copy of the content. In addition, it ensures embedding of a fingerprint even if a malicious user succeeds in extracting the decryption keys of a receiver. Furthermore, as long as the number of colluding users remains below a threshold, the colluders can only create decryption keys and content copies that incriminate at least one of them.

This paper enhances our extended abstract [1] in the following aspects. First, the extended abstract does not contain the security proof, which is the major contribution. Second, we show here that our instantiation of SSW is exact, whereas the extended abstract only claims this result. Last, we discuss here the trade-off between storage size and computation cost at the receiving end.

2 Related Work

The original Chameleon cipher of Anderson and Manifavas is 3-collusion-resistant [12]: A collusion of up to 3 malicious users has a negligible chance of creating a good copy that does not incriminate them. Each legitimate user knows the seed of a Pseudo-Random Sequence (PRS) and a long table filled with random keywords. Based on the sender's master table, each receiver obtains a slightly different table copy, where individual bits in the keywords are modified in a characteristic way. Interpreting the PRS as a sequence of addresses in the table, the sender adds the corresponding keywords in the master table bitwise modulo 2 in order to mask the plaintext word. The receiver applies the same operation to the ciphertext using its table copy, thus embedding the fingerprint.

The original cipher, however, has some inconveniences. Most importantly, it has no formal security analysis and bounds the collusion resistance by the constant number 3, whereas our scheme allows to choose this bound depending on the number of available watermark coefficients. In addition, the original scheme

[1] Our scheme does not yet support fingerprints based on coding theory.

limits the content space (and keywords) to strings with characteristic bit positions that may be modified without visibly altering the content. In contrast, our scheme uses algebraic operations in a group of large order, which enables modification of any bit in the keyword and processing of arbitrary documents.

Chameleon was inspired by work from Maurer [17,18]. His cipher achieves information-theoretical security in the bounded storage model with high probability. In contrast, Chameleon and our proposed scheme only achieve computational security. The reason is that the master table length in Maurer's cipher is super-polynomial. As any adversary would need to store most of the table to validate guesses, the bounded storage capacity defeats all attacks with high probability. However, Maurer's cipher was never intended to provide traceability of content or renewability, but only confidentiality.

Ferguson et al. discovered security weaknesses in a randomized stream cipher similar to Chameleon [19]. However, their attack only works for linear sequences of keywords in the master table, not for the PRSs of our proposed solution.

Ergun, Kilian, and Kumar prove that an averaging attack with additional Gaussian noise defeats any watermarking scheme [20]. Their bound on the minimum number of different content copies needed for the attack asymptotically coincides with the bound on the maximum number of different content copies to which the watermarking scheme of Kilian et al. is collusion-resistant [15]. As we can emulate [15] with our fingercasting approach, its collusion resistance is—at least asymptotically—the best we can hope for.

Recently there was a great deal of interest in joint fingerprinting and decryption [13,21,22,11,23]. Basically, we can distinguish three strands of work. The first strand of work applies Chameleon in different application settings. Briscoe et al. introduce Nark, which is an application of the original Chameleon scheme in the context of Internet multicast [13]. However, in contrast to our new Chameleon cipher they neither enhance Chameleon nor analyze its security. The second strand of work tries to achieve joint fingerprinting and decryption by either trusting network nodes to embed fingerprints (Watercasting in [21]) or doubling the size of the ciphertext by sending differently fingerprinted packets of content [22]. Our proposed solution neither relies on trusted network nodes nor increases the ciphertext size. The third strand of work proposes new joint fingerprinting and decryption processes, but at the price of replacing encryption with scrambling, which does not achieve indistinguishability of ciphertext and has security concerns [11,23]. In contrast, our new Chameleon cipher achieves indistinguishability of ciphertext.

3 Preliminaries

3.1 Notation

We recall some standard notations that will be used throughout the paper. First, we denote scalar objects with lower-case variables, e.g., o_1, and object tuples as

well as roles with upper-case variables, e.g., X_1. When we summarize objects or roles in set notation, we use an upper-case calligraphic variable, e.g., $\mathcal{O} := \{o_1, o_2, \ldots\}$ or $\mathcal{X} := \{X_1, X_2, \ldots\}$. Second, let A be an algorithm. By $y \leftarrow \mathsf{A}(x)$ we denote that y was obtained by running A on input x. If A is deterministic, then y is a variable with a unique value. Conversely, if A is probabilistic, then y is a random variable. For example, by $y \leftarrow \mathsf{N}(\mu, \sigma)$ we denote that y was obtained by selecting it at random with normal distribution, where μ is the mean and σ the standard deviation. Third, $o_1 \xleftarrow{R} \mathcal{O}$ and $o_2 \xleftarrow{R} [0, z]$ denote the selection of a random element of the set $\mathcal{O}$ and the interval $[0, z]$ with uniform distribution. Finally, $V \cdot W$ denotes the dot product of two vectors $V := (v_1, \ldots, v_n)$ and $W := (w_1, \ldots, w_n)$, which is defined as $V \cdot W := \sum_{j=1}^{n} v_j w_j$, while $||V||$ denotes the Euclidean norm $||V|| := \sqrt{V \cdot V}$.

3.2 Roles and Objects in Our System Model

The *(broadcast) center* manages the broadcast channel, distributes decryption keys and is fully trusted. The *users* obtain the content via devices that we refer to as *receivers*. For example, a receiver may be a set-top box in the context of pay-TV or a DVD player in movie distribution. We denote the number of receivers with N; the set of receivers is $\mathcal{U} := \{u_i \mid 1 \leq i \leq N\}$. When a receiver violates the terms and conditions of the application, e.g., leaks its keys or shares content, the center revokes the receiver's keys and thus makes them useless for decryption purposes. We denote the set of revoked receivers with $\mathcal{R} := \{r_1, r_2, \ldots\} \subset \mathcal{U}$.

We represent broadcast content as a sequence $M := (m_1, \ldots, m_n)$ of real numbers in $[0, z]$, where M is an element of the content space $\mathcal{M}$.[2] For example, these numbers may be the n most significant coefficients of the Discrete Cosine Transform (DCT) as described in [14]. However, they should not be thought of as a literal description of the underlying content, but as a representation of the values that are to be changed by the watermarking process [20]. We refer to these values as *significant* and to the remainder as *insignificant*. In the remainder of this paper, we only refer to the significant part of the content, but briefly comment on the insignificant part in Section 5.

3.3 Cryptographic Building Blocks

Negligible Function. A negligible function $\mathsf{f} : \mathbb{N} \to \mathbb{R}$ is a function where the inverse of any polynomial is asymptotically an upper bound:

$$\forall k > 0 \; \exists \lambda_0 \; \forall \lambda > \lambda_0 : \quad \mathsf{f}(\lambda) < 1/\lambda^k$$

Probabilistic Polynomial Time. A probabilistic polynomial-time algorithm is an algorithm for which there exists a polynomial poly such that for every input $x \in \{0,1\}^*$ the algorithm always halts after $\mathsf{poly}(|x|)$ steps, independently of the outcome of its internal coin tosses.

[2] Although this representation mainly applies to images, we discuss an extension to movies and songs in Section 5.

Pseudo-Random Sequence (PRS). We first define the notion of pseudo-randomness and then proceed to define a Pseudo-Random Sequence Generator (PRSG). For further details we refer to [24, Section 3.3.1]:

Definition 1 (Pseudo-randomness). *Let* $\mathsf{len} : \mathbb{N} \to \mathbb{N}$ *be a polynomial such that* $\mathsf{len}(\lambda) > \lambda$ *for all* $\lambda \in \mathbb{N}$ *and let* $U_{\mathsf{len}(\lambda)}$ *be a random variable uniformly distributed over the strings* $\{0,1\}^{\mathsf{len}(\lambda)}$ *of length* $\mathsf{len}(\lambda)$*. Then the random variable* X *with* $|X| = \mathsf{len}(\lambda)$ *is called* pseudo-random *if for every probabilistic polynomial-time distinguisher* $\mathcal{D}$*, the advantage* $\mathsf{Adv}(\lambda)$ *is a negligible function:*

$$\mathsf{Adv}(\lambda) := \left|\mathsf{Pr}\left[\mathcal{D}(X) = 1\right] - \mathsf{Pr}\left[\mathcal{D}(U_{\mathsf{len}(\lambda)}) = 1\right]\right|$$

Definition 2 (Pseudo-Random Sequence Generator). *A* PRSG *is a deterministic polynomial-time algorithm* G *that satisfies two requirements:*

1. *Expansion: There exists a polynomial* $\mathsf{len} : \mathbb{N} \to \mathbb{N}$ *such that* $\mathsf{len}(\lambda) > \lambda$ *for all* $\lambda \in \mathbb{N}$ *and* $|\mathsf{G}(str)| = \mathsf{len}(|str|)$ *for all* $str \in \{0,1\}^*$.
2. *Pseudo-randomness: The random variable* $\mathsf{G}(U_\lambda)$ *is pseudo-random.*

A PRS *is a sequence* $\mathsf{G}(str)$ *derived from a uniformly distributed random seed* str *using a PRSG.*

Chameleon Encryption. To set up a Chameleon scheme $\mathcal{CE} := (\mathsf{KeyGenCE}, \mathsf{KeyExtrCE}, \mathsf{EncCE}, \mathsf{DecCE}, \mathsf{DetectCE})$, the center generates the secret master table MT, the secret table fingerprints $TF := (TF^{(1)}, \ldots, TF^{(N)})$, and selects a threshold t using the key generation algorithm $(MT, TF, t) \leftarrow \mathsf{KeyGenCE}(N, 1^{\lambda'}, par_{\mathsf{CE}})$, where N is the number of receivers, λ' a security parameter, and par_{CE} a set of performance parameters. To add receiver u_i to the system, the center uses the key extraction algorithm $RT^{(i)} \leftarrow \mathsf{KeyExtrCE}(MT, TF, i)$ to deliver the secret receiver table $RT^{(i)}$ to u_i. To encrypt content M exclusively for the receivers in possession of a receiver table $RT^{(i)}$ and a fresh session key k^{sess}, the center uses the encryption algorithm $C \leftarrow \mathsf{EncCE}(MT, k^{\mathsf{sess}}, M)$, where the output is the ciphertext C. Only a receiver u_i in possession of $RT^{(i)}$ and k^{sess} is capable of decrypting C and obtaining a fingerprinted copy $M^{(i)}$ of content M using the decryption algorithm $M^{(i)} \leftarrow \mathsf{DecCE}(RT^{(i)}, k^{\mathsf{sess}}, C)$.

When the center discovers an illegal copy M^* of content M, it executes $\mathsf{DetectCE}$, which uses the fingerprint detection algorithm $\mathsf{DetectFP}$ of the underlying fingerprinting scheme to detect whether $RT^{(i)}$ left traces in M^*. For further details on our notation of a Chameleon scheme, we refer to Appendix C.

Fingerprinting. To set up a fingerprinting scheme, the center generates the secret content fingerprints $CF := (CF^{(1)}, \ldots, CF^{(N)})$ and the secret similarity threshold t using the setup algorithm $(CF, t) \leftarrow \mathsf{SetupFP}(N, n', par_{\mathsf{FP}})$, where N is the number of receivers, n' the number of content coefficients, and par_{FP} a set of performance parameters. To embed the content fingerprint $CF^{(i)} := (cf_1^{(i)}, \ldots, cf_{n'}^{(i)})$ of receiver u_i into the original content M, the center uses the embedding algorithm $M^{(i)} \leftarrow \mathsf{EmbedFP}(M, CF^{(i)})$. To verify whether an illegal copy M^* of content M contains traces of the content fingerprint $CF^{(i)}$ of receiver

u_i, the center uses the detection algorithm $dec \leftarrow \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)$. It calculates the similarity between the detected fingerprint $CF^* := M^* - M$ and $CF^{(i)}$ using a similarity measure. If the similarity is above the threshold t, then the center declares u_i guilty ($dec = \texttt{true}$), otherwise innocent ($dec = \texttt{false}$). This type of detection algorithm is called non-blind because it needs the original content M as input; the opposite is a blind detection algorithm.

We call a fingerprinting scheme *additive* if the probability distribution ProDis of its coefficients has the following property: Adding two independent random variables that follow ProDis results in a random variable that also follows ProDis. For example, the normal distribution has this property, where the means and variances add up during addition.

Spread Spectrum Watermarking (SSW) is an instance of an additive fingerprinting scheme. We describe the SSW scheme of [15], which we later use to achieve collusion resistance. The content fingerprint $CF^{(i)}$ consists of independent random variables $cf_j^{(i)}$ with normal distribution $\mathsf{ProDis} = \mathsf{N}(0, \sigma')$, where σ' is a function $\mathsf{f}_{\sigma'}(N, n', par_{\mathsf{FP}})$. The similarity threshold t is a function $\mathsf{f}_t(\sigma', N, par_{\mathsf{FP}})$. Both functions $\mathsf{f}_{\sigma'}$ and f_t are specified in [15]. During $\mathsf{EmbedFP}$, the center adds the fingerprint coefficients to the content coefficients: $m_j^{(i)} \leftarrow m_j + cf_j^{(i)}$. The similarity test is $\mathsf{Sim}(CF^*, CF^{(i)}) \geq t$ with $\mathsf{Sim}(CF^*, CF^{(i)}) := (CF^* \cdot CF^{(i)})/||CF^*||$. Finally, the scheme's security is given by:

Theorem 1. [15, Section 3.4] *In the SSW scheme with the above parameters, an adversarial coalition needs* $\Omega(\sqrt{n'/\ln N})$ *differently fingerprinted copies of content* M *to have a non-negligible chance of creating a good copy* M^* *without any coalition member's fingerprint.*

For further details on our notation of a fingerprinting scheme and the SSW scheme of [15], we refer to Appendix D.

Broadcast Encryption. To set up the scheme, the center generates the secret master key MK using the key generation algorithm $MK \leftarrow \mathsf{KeyGenBE}(N, 1^{\lambda''})$, where N is the number of receivers and λ'' the security parameter. To add receiver u_i to the system, the center uses the key extraction algorithm $SK^{(i)} \leftarrow \mathsf{KeyExtrBE}(MK, i)$ to extract the secret key $SK^{(i)}$ of u_i. To encrypt session key k^{sess} exclusively for the non-revoked receivers $\mathcal{U} \setminus \mathcal{R}$, the center uses the encryption algorithm $C \leftarrow \mathsf{EncBE}(MK, \mathcal{R}, k^{\mathsf{sess}})$, where the output is the ciphertext C. Only a non-revoked receiver u_i has a matching private key $SK^{(i)}$ that allows to decrypt C and obtain k^{sess} using the decryption algorithm $k^{\mathsf{sess}} \leftarrow \mathsf{DecBE}(i, SK^{(i)}, C)$. For further details on our notation of a BE scheme, we refer to Appendix E.

3.4 Requirements of a Fingercasting Scheme

Before we enter into the details of our fingercasting approach, we summarize its requirements: correctness, security, collusion resistance, and frame-proofness. To put it simply, the aim of our fingercasting approach is to generically combine an instance of a BE scheme, a Chameleon scheme, and a fingerprinting scheme

such that the combination inherits the security of BE and Chameleon as well as the collusion resistance of fingerprinting. To define correctness we first need to clarify how intrusive a fingerprint may be. For a copy to be good, the fingerprint may not perceptibly deteriorate its quality:

Definition 3 (Goodness). Goodness *is a predicate* $\mathsf{Good} : \mathcal{M}^2 \rightarrow \{\mathtt{true}, \mathtt{false}\}$ *over two messages* $M_1, M_2 \in \mathcal{M}$ *that evaluates their perceptual difference. A fingerprinted copy* $M^{(i)}$ *is called* good *if its perceptual difference to the original content* M *is below a perceptibility threshold. We denote this with* $\mathsf{Good}(M^{(i)}, M) = \mathtt{true}$*. Otherwise, the copy is called* bad.

Definition 4 (Correctness). *Let* $p^{\mathsf{bad}} \ll 1$ *be the maximum allowed probability of a bad copy. A fingercasting scheme is* correct *if the probability for a non-revoked receiver to obtain a bad copy* $M^{(i)}$ *of the content* M *is at most* p^{bad}*, where the probability is taken over all coin tosses of the setup and encryption algorithm:*

$$\forall M \in \mathcal{M}, \forall u_i \in \mathcal{U} \setminus \mathcal{R}: \quad \mathsf{Pr}\left[\mathsf{Good}(M, M^{(i)}) = \mathtt{false}\right] \leq p^{\mathsf{bad}}$$

The SSW scheme of [15] uses the goodness predicate $||M^{(i)} - M|| \leq \sqrt{n'}\delta$, where n' is the number of content coefficients and δ a goodness criterion.

All relevant BE schemes provide IND-CCA1 security [6,7,8], which is a stronger notion than IND-CPA security. As we aim to achieve at least IND-CPA security, the remaining requirements only relate to the Chameleon scheme $\mathcal{CE}$.

We define IND-CPA security of $\mathcal{CE}$ by a game between an IND-CPA adversary $\mathcal{A}$ and a challenger $\mathcal{C}$: The challenger runs $(MT, TF, t) \leftarrow \mathsf{KeyGenCE}(N, 1^{\lambda'}, par_{\mathsf{CE}})$, generates a secret random session key k^{sess} and sends (MT, TF, t) to $\mathcal{A}$. $\mathcal{A}$ outputs two content items $M_0, M_1 \in \mathcal{M}$ on which it wishes to be challenged. $\mathcal{C}$ picks a random bit $b \overset{R}{\leftarrow} \{0,1\}$ and sends the challenge ciphertext $C_b \leftarrow \mathsf{EncCE}(MT, k^{\mathsf{sess}}, M_b)$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess b' and wins if $b' = b$. We define the advantage of $\mathcal{A}$ against $\mathcal{CE}$ as $\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A},\mathcal{CE}}(\lambda') := |\mathsf{Pr}[b' = 0 | b = 0] - \mathsf{Pr}[b' = 0 | b = 1]|$. For further details on security notions we refer to [25].

Definition 5 (IND-CPA security). *A Chameleon scheme* $\mathcal{CE}$ *is* IND-CPA secure *if for every probabilistic polynomial-time* IND-CPA *adversary* $\mathcal{A}$ *we have that* $\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A},\mathcal{CE}}(\lambda')$ *is a negligible function.*

We note that in Definition 5, the adversary is not an outsider or third party, but an insider in possession of the master table (not only a receiver table). Nevertheless, the adversary should have a negligible advantage in distinguishing the ciphertexts of two messages of his choice as long as the session key remains secret.

Collusion resistance is defined by the following game between an adversarial coalition $\mathcal{A} \subseteq \mathcal{U} \setminus \mathcal{R}$ and a challenger $\mathcal{C}$: The challenger runs KeyGenCE on parameters $(N, 1^{\lambda'}, par_{\mathsf{CE}})$, generates a ciphertext $C \leftarrow \mathsf{EncCE}(MT, k^{\mathsf{sess}}, M)$, and gives $\mathcal{A}$ the receiver tables $RT^{(i)}$ of all coalition members as well as the session key k^{sess} . Then $\mathcal{A}$ outputs a document copy M^* and wins if for all coalition members the detection algorithm fails (false negative):

Definition 6 (Collusion resistance). *Let* $\mathsf{DetectFP}$ *be the fingerprint detection algorithm of the fingerprinting scheme that a Chameleon scheme* $\mathcal{CE}$ *instantiates. Then* $\mathcal{CE}$ *is* (q, p^{neg})-collusion-resistant *if for every probabilistic polynomial-time adversarial coalition* $\mathcal{A}$ *of at most* $q := |\mathcal{A}|$ *colluders we have that*

$$\Pr\left[\mathsf{Good}(M^*, M)=\mathtt{true},\, \forall u_i \in \mathcal{A} : \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)=\mathtt{false}\right] \leq p^{\mathsf{neg}},$$

where the false negative probability is taken over the coin tosses of the setup algorithm, of the adversarial coalition $\mathcal{A}$*, and of the session key* k^{sess}*.*

Note that 1-collusion resistance is also called robustness. Frame-proofness is similar to collusion resistance, but $\mathcal{A}$ wins the game if the detection algorithm accuses an innocent user (false positive).

Definition 7 (Frame-proofness). *Let* $\mathsf{DetectFP}$ *be the fingerprint detection algorithm of the fingerprinting scheme that a Chameleon scheme* $\mathcal{CE}$ *instantiates. Then* $\mathcal{CE}$ *is* (q, p^{pos})-frame-proof *if for every probabilistic polynomial-time adversarial coalition* $\mathcal{A}$ *of at most* $q := |\mathcal{A}|$ *colluders we have that*

$$\Pr\left[\mathsf{Good}(M^*, M)=\mathtt{true},\, \exists u_i \notin \mathcal{A} : \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)=\mathtt{true}\right] \leq p^{\mathsf{pos}},$$

where the false positive probability is taken over the coin tosses of the setup algorithm, of the adversarial coalition $\mathcal{A}$*, and of the session key* k^{sess}*.*

In Definitions 6 and 7, the adversarial coalition again consists of insiders in possession of their receiver tables and the session key. Nevertheless, the coalition should have a well-defined and small chance of creating a plaintext copy that incriminates none of the coalition members (collusion resistance) or an innocent user outside the coalition (frame-proofness).

4 Proposed Solution

4.1 High-Level Overview of the Proposed Fingercasting Scheme

To fingercast content, the center uses the BE scheme to send a fresh session key to each non-revoked receiver. This session key initializes a pseudo-random sequence generator. The resulting pseudo-random sequence represents a sequence of addresses in the master table of our new Chameleon scheme. The center encrypts the content with the master table entries to which the addresses refer. Each receiver has a unique receiver table that differs only slightly from the master table. During decryption, these slight differences in the receiver table lead to slight, but characteristic differences in the content copy.

Interaction Details. We divide this approach into the same five steps that we have seen for Chameleon schemes in Section 3.3. First, the *key generation* algorithm of the fingercasting scheme consists of the key generations algorithms of the two underlying schemes $\mathsf{KeyGenBE}$ and $\mathsf{KeyGenCE}$. The center's master key thus consists of MK, MT and TF. Second, the same observation holds

for the *key extraction* algorithm of the fingercasting scheme. It consists of the respective algorithms in the two underlying schemes KeyExtrBE and KeyExtrCE. The secret key of receiver u_i therefore has two elements: $SK^{(i)}$ and $RT^{(i)}$.

Third, the *encryption* algorithm defines how we interlock the two underlying schemes. To encrypt, the center generates a fresh and random session key $k^{\mathsf{sess}} \xleftarrow{R} \{0,1\}^{\lambda}$. This session key is broadcasted to the non-revoked receivers using the BE scheme: $C_{\mathsf{BE}} \leftarrow \mathsf{EncBE}(MK, \mathcal{R}, k^{\mathsf{sess}})$. Subsequently, the center uses k^{sess} to determine addresses in the master table MT of the Chameleon scheme and encrypts with the corresponding entries: $C_{\mathsf{CE}} \leftarrow \mathsf{EncCE}(MT, k^{\mathsf{sess}}, M)$. The ciphertext of the fingercasting scheme thus has two elements C_{BE} and C_{CE}.

Fourth, the *decryption* algorithm inverts the encryption algorithm with unnoticeable, but characteristic errors. First of all, each non-revoked receiver u_i recovers the correct session key: $k^{\mathsf{sess}} \leftarrow \mathsf{DecBE}(i, SK^{(i)}, C_{\mathsf{BE}})$. Therefore, u_i can recalculate the PRS and the correct addresses in receiver table $RT^{(i)}$. However, this receiver table is slightly different from the master table. Therefore, u_i obtains a fingerprinted copy $M^{(i)}$ that is slightly different from the original content: $M^{(i)} \leftarrow \mathsf{DecCE}(RT^{(i)}, k^{\mathsf{sess}}, C_{\mathsf{CE}})$. Last, the *fingerprint detection* algorithm of the fingercasting scheme is identical to that of the underlying fingerprinting scheme.

4.2 A New Chameleon Scheme

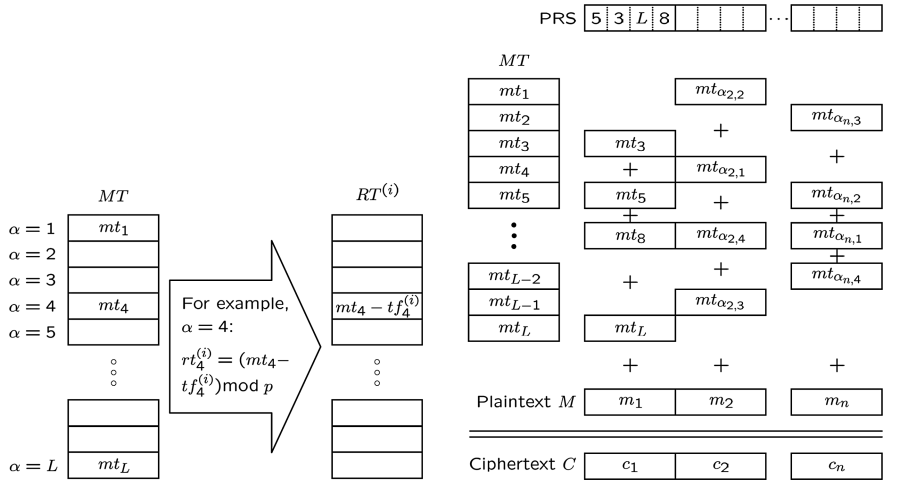
Up to now, we have focused on the straightforward aspects of our approach; we have neglected the intrinsic difficulties and the impact of the requirements on the Chameleon scheme. In the sequel, we will show a specific Chameleon scheme that fulfills all of them. We design it in such a way that its content fingerprints can emulate any additive fingerprinting scheme, which we later instantiate with the SSW scheme as proof of concept.

Key Generation. To define this algorithm, we need to determine how the center generates the master table MT and the table fingerprints TF. To generate MT, the center chooses L table entries at random from the interval $[0, z]$ with independent uniform distribution: $mt_\alpha \xleftarrow{R} [0, z]$ for all $\alpha \in \{1, \ldots, L\}$. As the table entries will be addressed with bit words, we select $L = 2^l$ such that l indicates the number of bits needed to define the binary address of an entry in the table. The center thus obtains the master table $MT := (mt_1, mt_2, \ldots, mt_L)$.

To generate the table fingerprints $TF := (TF^{(1)}, \ldots, TF^{(N)})$, the center selects for each receiver u_i and each master table entry mt_α a fingerprint coefficient in order to disturb the original entry. Specifically, each fingerprint coefficient $tf_\alpha^{(i)}$ of table fingerprint $TF^{(i)}$ is independently distributed according to the probability distribution ProDis of the additive fingerprinting scheme, but scaled down with an attenuation factor $f \in \mathbb{R}$, $f \geq 1$:

$$tf_\alpha^{(i)} \leftarrow 1/f \cdot \mathsf{ProDis}(par_{\mathsf{FP}}) \tag{1}$$

Key Extraction. After the probabilistic key generation algorithm we now describe the deterministic key extraction algorithm. The center processes table



(a) To derive $RT^{(i)}$ from MT, the center subtracts the L fingerprint coefficients $tf_\alpha^{(i)}$ at address α for all $\alpha \in \{1, \ldots, L\}$.

(b) To derive ciphertext C from plaintext M, the center uses the session key to generate a PRS. It then adds the addressed master table entries to the plaintext.

Fig. 1. Receiver table derivation and ciphertext calculation

fingerprint $TF^{(i)} := (tf_1^{(i)}, \ldots, tf_L^{(i)})$ of receiver u_i as follows: The center subtracts each fingerprint coefficient in $TF^{(i)}$ from the corresponding master table entry to obtain the receiver table entry, which we illustrate in Fig. 1(a):

$$\forall\, \alpha \in \{1, \ldots, L\} : \quad rt_\alpha^{(i)} \leftarrow mt_\alpha - tf_\alpha^{(i)} \bmod p \tag{2}$$

Remark 1. The modulo operator allows only integer values to be added. However, the master table, the table fingerprints and the content coefficients are based on real numbers with finite precision. We solve this ostensible contradiction by scaling the real values to the integer domain by an appropriate scaling factor ρ, possibly ignoring further decimal digits. ρ must be chosen large enough to allow a computation in the integer domain with a sufficiently high precision. We implicitly assume this scaling to the integer domain whenever real values are used. For example, with real-valued variables $rt^{(i)}$, mt, and $tf^{(i)}$ the operation $rt^{(i)} \leftarrow (mt - tf^{(i)}) \bmod p$ actually stands for $\rho \cdot rt^{(i)} \leftarrow (\rho \cdot mt - \rho \cdot tf^{(i)}) \bmod p$. The group order $p := \lceil \rho \cdot z \rceil + 1$ is defined by the content space $[0, z]$ (see Section 3.2) and the scaling factor ρ.

Encryption. Fig. 1(b) gives an overview of the encryption algorithm. The session key k^{sess} is used as the seed of a PRSG with expansion function $\mathsf{len}(|k^{\mathsf{sess}}|) \geq n \cdot s \cdot l$, where parameter s will be specified below. To give a practical example for a PRSG, k^{sess} may serve as the key for a conventional block cipher, e.g., AES or

triple DES,[3] in output feedback mode. Each block of l bits of the pseudo-random sequence is interpreted as an address β in the master table MT. For each coefficient of the plaintext, the center uses s addresses that define s entries of the master table. In total, the center obtains $n \cdot s$ addresses that we denote with $\beta_{j,k}$, where j is the coefficient index, k the address index, and $\mathsf{Extract}_i$ extracts the i-th block of length l from its input string:

$$\forall j \in \{1, \ldots, n\},\ \forall k \in \{1, \ldots, s\} : \quad \beta_{j,k} \leftarrow \mathsf{Extract}_{(j-1)s+k}(\mathsf{G}(k^{\mathsf{sess}})) \tag{3}$$

For each content coefficient, the center adds the s master table entries modulo the group order. In Fig. 1(b), we illustrate the case $s = 4$, which is the design choice in the original Chameleon cipher. The j-th coefficient c_j of the ciphertext C is calculated as

$$\forall j \in \{1, \ldots, n\} : \quad c_j \leftarrow \left(m_j + \sum_{k=1}^{s} mt_{\beta_{j,k}} \right) \bmod p\,, \tag{4}$$

where $mt_{\beta_{j,k}}$ denotes the master table entry referenced by address $\beta_{j,k}$ from (3).

Decryption. The decryption algorithm proceeds in the same way as the encryption algorithm with two exceptions. First, the receiver has to use its receiver table $RT^{(i)}$ instead of MT. Second, the addition is replaced by subtraction. The j-th coefficient $m_j^{(i)}$ of the plaintext copy $M^{(i)}$ of receiver u_i is thus calculated as

$$m_j^{(i)} \leftarrow \left(c_j - \sum_{k=1}^{s} rt_{\beta_{j,k}}^{(i)} \right) \bmod p, \tag{5}$$

where $rt_{\beta_{j,k}}^{(i)}$ denotes the receiver table entry of receiver u_i referenced by address $\beta_{j,k}$ generated in (3). As the receiver table $RT^{(i)}$ slightly differs from the master table MT, the plaintext copy $M^{(i)}$ obtained by receiver u_i slightly differs from the original plaintext M. By appropriately choosing the attenuation factor f in (1), the distortion of $M^{(i)}$ with respect to M is the same as that of the instantiated fingerprinting scheme and goodness is preserved (see Section 4.3).

Fingerprint Detection. When the center detects an illegal copy $M^* = (m_1^*, \ldots, m_n^*)$ of content M, it tries to identify the receivers that participated in the generation of M^*. To do so, the center verifies whether the fingerprint of a suspect receiver u_i is present in M^*. Obviously, the fingerprint is unlikely to appear in its original form; an adversary may have modified it by applying common attacks such as resampling, requantization, compression, cropping, and rotation. Furthermore, the adversary may have applied an arbitrary combination of these known attacks and other yet unknown attacks. Finally, an adversarial coalition may have colluded and created M^* using several different copies of M.

The fingerprint detection algorithm is identical to that of the underlying fingerprinting scheme: $dec \leftarrow \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)$. In order to properly scale

[3] Advanced Encryption Standard [26] and Data Encryption Standard [27].

the content fingerprint, we need to select the attenuation factor f in (1). We choose it such that the addition of s attenuated fingerprint coefficients generates a random variable that follows ProDis *without* attenuation (for an example see Section 4.3). In order to verify whether the table fingerprint $TF^{(i)}$ of receiver u_i left traces in M^*, DetectFP calculates the similarity between the detected content fingerprint CF^* with coefficients $cf_j^* := m_j^* - m_j$ and the content fingerprint $CF^{(i)}$ in u_i's copy $M^{(i)}$ with

$$cf_j^{(i)} := m_j^{(i)} - m_j \overset{(4),(5)}{=} \sum_{k=1}^{s} \left(mt_{\beta_{j,k}} - rt_{\beta_{j,k}}^{(i)} \right) \overset{(2)}{=} \sum_{k=1}^{s} tf_{\beta_{j,k}}^{(i)}, \tag{6}$$

where $tf_{\beta_{j,k}}^{(i)}$ is the fingerprint coefficient that fingerprinted receiver table $RT^{(i)}$ at address $\alpha = \beta_{j,k}$ in (2). If the similarity is above threshold t, the center declares u_i guilty. Note that the calculation of CF^* necessitates the original content M, whereas the calculation of $CF^{(i)}$ relies on the session key k^{sess} and the table fingerprint $TF^{(i)}$; the scheme is thus non-blind in its current version. However, we assume it is possible to design an extended scheme with a blind detection algorithm. If instantiated with Spread Spectrum Watermarking, the watermark is often robust enough to be detected even in the absence of the original content.

The same algorithm applies to detection of fingerprints in illegal copies of receiver tables. Their fingerprints have the same construction and statistical properties, where the attenuated amplitude of the fingerprint coefficients in (1) is compensated by a higher number of coefficients, as the relation $L/f \approx n$ holds for practical parameter choices (see Section 5.1).

When the center detects the fingerprint of a certain user in an illegal content copy or an illegal receiver table, there are two potential countermeasures with different security and performance tradeoffs. One is to simply revoke the user in the BE scheme such that the user's BE decryption key becomes useless and no longer grants access to the session key. However, the user's receiver table still allows to decrypt content if yet another user illegally shares the session key. In an Internet age, this is a valid threat as two illegal users may collude such that one user publishes the receiver table (and gets caught) and the other user anonymously publishes the session keys (and doesn't get caught). Nevertheless, we stress that this weakness, namely the non-traceability of session keys, is common to all revocation BE schemes because the session key is identical for all users and therefore does not allow tracing.[4]

In order to avoid this weakness, the other potential countermeasure is to not only revoke the user whose receiver table was illegally shared, but also renew the master table and redistribute the new receiver tables. If the broadcast channel has enough spare bandwidth, the center can broadcast the receiver tables individually to all receivers in off-peak periods, i.e., when the channel's bandwidth

[4] The common assumption for revocation BE schemes is that it is difficult to share the session key anonymously on a large scale without being caught. Even if key sharing may be possible on a small scale, e.g., among family and friends, the main goal is to allow revocation of a user that shared the decryption key or session keys and got caught, no matter by which means of technical or legal tracing.

is not fully used for regular transmission. The relevant BE schemes [6,7,8] allow to encrypt each receiver table individually for the corresponding receiver such that only this receiver can decrypt and obtain access.[5] If the broadcast channel's bandwidth is too low, then the receiver tables need to be redistributed as in the initial setup phase, e.g., via smartcards.

Parameter Selection. The new Chameleon scheme has two major parameters L and s that allow a trade-off between the size of $RT^{(i)}$, which u_i has to store, and the computation cost, which grows linearly with the number s of addresses per content coefficient in (4). By increasing L, we can decrease s in order to replace computation cost with storage size. Further details follow in Section 5.1.

4.3 Instantiation with Spread Spectrum Watermarking

In this section, we instantiate the fingerprinting scheme with the SSW scheme of [15] and thereby inherit its collusion resistance and frame-proofness. Let the center choose the SSW scheme's parameters $par_{\mathsf{FP}} = (\delta, p^{\mathsf{bad}}, p^{\mathsf{pos}})$, which allows to calculate a standard deviation σ' and a threshold t via two functions $\mathsf{f}_{\sigma'}(N, n', \delta, p^{\mathsf{bad}})$ and $\mathsf{f}_t(\sigma', N, p^{\mathsf{pos}})$ defined in [15]. The probability distribution of the SSW scheme is then $\mathsf{ProDis} = \mathsf{N}(0, \sigma')$. We set $f = s$ because then $1/f \cdot \mathsf{N}(0, \sigma')$ in (1) is still a normal distribution with mean 0 and standard deviation $1/\sqrt{s} \cdot \sigma'$, and adding s of those variables in (4) and (5) leads to the required random variable with standard deviation σ'. It remains to define the similarity measure for the detection algorithm $dec \leftarrow \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)$, which [15] defines as:

$$dec = \mathtt{true} \quad \text{if} \quad \frac{CF^* \cdot CF^{(i)}}{||CF^*||} > t$$

We call an instantiation *exact* if it achieves the same statistical properties as the fingerprinting scheme that it instantiates. Theorem 2 below states that the above choice is an exact instantiation of the SSW scheme.

Theorem 2. *Let σ' and σ be the standard deviations of the SSW scheme and the Chameleon scheme instantiated with SSW, respectively, and n' and n be their number of content coefficients. Then the following mapping between both schemes is an exact instantiation:*

$$\sigma' = \sqrt{s} \cdot \sigma \ (\Leftrightarrow f = s) \qquad \textit{and} \qquad n' = n$$

Towards the proof of Theorem 2. We prove an even stronger result than Theorem 2. In addition to the exactness of the instantiation, we also prove that it is optimal to fingerprint *every* entry of the receiver tables. To do so, we first formulate Lemmata 1–4 and then describe why they imply Theorem 2. For

[5] In all of these schemes, the center shares with each user an individual secret, which they can use for regular symmetric encryption.

the Lemmata, we introduce a parameter $F \in \{1, 2, \ldots, L\}$ that describes the number of receiver table entries that obtain a fingerprint coefficient $tf_{\alpha}^{(i)}$ in (2). The position of the F fingerprinted entries in the receiver table is selected with uniform distribution. We show that the choice $F = L$ is optimal in the sense that the resulting instantiation is exact.

The difficulty in analyzing the SSW instantiation is that each content coefficient is not only fingerprinted with a single fingerprint coefficient as in SSW, but with up to s such variables as can be seen from (6). Note that for $F < L$ some receiver table entries do not receive a fingerprint coefficient and are therefore identical to the master table entry. In order to analyze the statistical properties of the resulting fingerprint, we will need to calculate the expectation and variance of two parameters that link the instantiation to the original SSW scheme.

The first parameter is the number N^{fp} of fingerprint coefficients $tf^{(i)}$ that are added to a content coefficient m_j by using the receiver table $RT^{(i)}$ in (5) instead of the master table MT in (4). In SSW, N^{fp} has the constant value 1, i.e., a content fingerprint consists of one fingerprint coefficient per content coefficient, whereas in our scheme N^{fp} varies between 0 and s as shown in (6). If only F of the L receiver table entries have been fingerprinted, then $tf^{(i)} = 0$ for the remaining $L - F$ entries.

The second parameter is the number of content coefficients that carry a detectable content fingerprint. In SSW, this number has the constant value n', i.e., every coefficient carries a fingerprint with fixed standard deviation, whereas in our scheme, some of the n coefficients may happen to receive no or only few fingerprint coefficients $tf^{(i)}$. Specifically, this happens when the receiver table entry $rt_{\beta_{j,k}}^{(i)}$ of (5) did not receive a fingerprint coefficient in (2) for $F < L$. The next lemma gives the number of normally distributed table fingerprint coefficients that our scheme adds to a content coefficient. This number is a random variable characterized by its expectation and standard variance.

We prove the lemmata under the *uniform sequence assumption*, i.e., the sequence used to select the addresses from the master table has independent uniform distribution. We stress that we only use it to find the optimal mapping with SSW; security and collusion resistance of the proposed scheme do *not* rely on this assumption for the final choice of parameters (see the end of this section).[6]

Lemma 1. *Let N^{fp} be the random variable counting the number of fingerprinted receiver table entries with which a coefficient $m_j^{(i)}$ of copy $M^{(i)}$ is fingerprinted. Then the probability of obtaining $k \in \{0, \ldots, s\}$ fingerprinted entries is*

$$\mathsf{Pr}\left[N^{\mathsf{fp}} = k\right] = \binom{s}{k} \left(\frac{F}{L}\right)^k \left(1 - \frac{F}{L}\right)^{s-k}$$

The expectation and the variance of N^{fp} are

$$\mathsf{E}(N^{\mathsf{fp}}) = s\frac{F}{L} \quad \textit{and} \quad \sigma^2_{N^{\mathsf{fp}}} := \mathsf{E}([N^{\mathsf{fp}} - \mathsf{E}(N^{\mathsf{fp}})]^2) = s\frac{F}{L}\left(1 - \frac{F}{L}\right).$$

[6] Note that even if this was not the case, we can show that the adversary's advantage is still negligible by a simple reduction argument.

Proof. During decryption, the receiver subtracts s receiver table entries $rt_\alpha^{(i)}$ from the ciphertext coefficient using (5). Each entry $rt_\alpha^{(i)}$ is either fingerprinted or not. Under the uniform sequence assumption, the addresses of the subtracted entries $rt_\alpha^{(i)}$ have independent uniform distribution. In addition, the F fingerprinted entries are distributed over $RT^{(i)}$ with independent uniform distribution. Therefore, the probability that a single address $\alpha = \beta_{j,k}$ in (5) points to a fingerprinted receiver table entry $rt_\alpha^{(i)}$ is F/L, which is the number of fingerprinted receiver table entries divided by the total number of entries. As the underlying experiment is a sequence of s consecutive yes-no experiments with success probability F/L, it follows that N^{fp} has binomial distribution. This implies the probability, the expectation, and the variance.

Lemma 1 allows us to determine how many fingerprint coefficients we can expect in each content coefficient and how the number of such fingerprint coefficients varies. The next question is what kind of random variable results from adding N^{fp} fingerprint coefficients.

Lemma 2. *By adding a number N^{fp} of independent $\mathsf{N}(0,\sigma)$-distributed fingerprint coefficients, the resulting random variable has normal distribution with mean 0 and standard deviation $\sqrt{N^{\mathsf{fp}}}\sigma$.*

Proof. Each fingerprint coefficient is independently distributed according to the normal distribution $\mathsf{N}(0,\sigma)$. When two independent and normally distributed random variables are added, the resulting random variable is also normally distributed, while the means and the variances add up. Due to linearity, the resulting standard deviation for N^{fp} random variables is $\sqrt{N^{\mathsf{fp}}\sigma^2} = \sqrt{N^{\mathsf{fp}}}\sigma$.

In order to fingerprint the content coefficients with the same standard deviation σ' as in the SSW scheme, the natural choice is to choose σ such that $\sqrt{\mathsf{E}(N^{\mathsf{fp}})}\sigma = \sigma'$. The remaining question is how many content coefficients are actually fingerprinted; note that due to the randomness of N^{fp}, some content coefficients may receive more fingerprint coefficients than others. We determine the expected number of fingerprinted content coefficients in the next two lemmata, while we leave it open how many fingerprint coefficients are needed for detection:

Lemma 3. *Let $N_{\mathsf{min}}^{\mathsf{fp}} \in \{1,\ldots,s\}$ be the minimum number of table fingerprint coefficients needed to obtain a detectable fingerprint in content coefficient $m_j^{(i)}$. Then the probability p^{fing} that coefficient $m_j^{(i)}$ of copy $M^{(i)}$ obtains at least $N_{\mathsf{min}}^{\mathsf{fp}}$ fingerprint coefficients is*

$$p^{\mathsf{fing}} = \sum_{k=N_{\mathsf{min}}^{\mathsf{fp}}}^{s} \binom{s}{k} (\frac{F}{L})^k (1-\frac{F}{L})^{s-k}$$

Proof. The lemma is a corollary of Lemma 1 by adding the probabilities of all events whose value of N^{fp} is greater than or equal to $N_{\mathsf{min}}^{\mathsf{fp}}$.

Lemma 4. *Let $N^{\mathsf{fing}} \in \{0, \ldots, n\}$ be the random variable counting the number of fingerprinted coefficients. Then the expectation of N^{fing} is*

$$\mathsf{E}(N^{\mathsf{fing}}) = \sum_{j=0}^{n} j \binom{n}{j} (p^{\mathsf{fing}})^j (1 - p^{\mathsf{fing}})^{n-j} = np^{\mathsf{fing}}$$

Proof. The lemma follows from the fact that N^{fing} has binomial distribution with success probability p^{fing} and n experiments.

Given Lemmata 1–4 we can derive some of the parameters in our scheme from SSW. Suppose that the center has already selected the parameters of the SSW scheme such that the requirements on the number of receivers and collusion resistance are met. This includes the choice of N, n', and $par_{\mathsf{FP}} = par_{\mathsf{CE}} := (\delta, p^{\mathsf{bad}}, p^{\mathsf{pos}})$; it allows to derive σ' and t of SSW based on the functions $\mathsf{f}_{\sigma'}(N, n', \delta, p^{\mathsf{bad}})$ and $\mathsf{f}_t(\sigma', N, p^{\mathsf{pos}})$, which are defined in [15].

Based on the center's selection, we can derive the parameters n, F/L, and $\sqrt{s} \cdot \sigma$ in our Chameleon scheme as follows. Our first aim is to achieve the same expected standard deviation in the content coefficients of our scheme as in SSW, i.e., $\sigma' = \sqrt{\mathsf{E}(N^{\mathsf{fp}})} \cdot \sigma$, which by Lemma 1 leads to $\sigma' = \sqrt{sF/L} \cdot \sigma$. Our second aim is to minimize the variance of N^{fp} in order to have $N^{\mathsf{fp}} = \mathsf{E}(N^{\mathsf{fp}})$ not only on average, but for as many content coefficients as possible, where $N^{\mathsf{fp}} = \mathsf{E}(N^{\mathsf{fp}})$ implies that the content coefficient in our scheme obtains a fingerprint with the same statistical properties as in SSW. The two minima of $\sigma^2_{N^{\mathsf{fp}}} = s \cdot F/L \cdot (1 - F/L)$ are $F/L = 0$ and $F/L = 1$, of which only the second is meaningful. $F/L = 1$ or $F = L$ is the case where all entries of the master table are fingerprinted. As this optimum case leads to a variance of $\sigma^2_{N^{\mathsf{fp}}} = 0$ and $N^{\mathsf{fp}} = s$, the content coefficients of our scheme and SSW have the same statistical properties. This proves Theorem 2 and the claim that all tables entries should be fingerprinted.

With $F/L = 1$ and $\sigma' = \sqrt{s} \cdot \sigma$, we obtain $\mathsf{Pr}\left[N^{\mathsf{fp}} = s\right] = 1$ by Lemma 1 and $p^{\mathsf{fing}} = 1$ by Lemma 3. Finally, we conclude that $\mathsf{E}(N^{\mathsf{fing}}) = n \cdot p^{\mathsf{fing}} = n$ by Lemma 4 and set $\mathsf{E}(N^{\mathsf{fing}}) = n = n'$. We stress that the equalities hold even if we replace the uniform sequence with a pseudo-random sequence; for $F = L$ the equations $N^{\mathsf{fp}} = s$ and $N^{\mathsf{fing}} = n$ are obviously independent of the uniform distribution of the sequence of addresses in the master table.

We note that the number s of addresses per content coefficient, introduced in (4), is still undetermined and may be chosen according to the security requirements (see Section 4.4).

4.4 Analysis

Correctness, Collusion Resistance and Frame-Proofness. Correctness follows from the correctness of the two underlying schemes, i.e., the BE scheme and the Chameleon scheme. Correctness of the Chameleon scheme follows from the correctness of the underlying fingerprinting scheme, which we can instantiate exactly by properly choosing the scaling factor in (1) and thus making the content fingerprint of (6) identical to a fingerprint of the instantiated fingerprinting

scheme. Collusion resistance and frame-proofness of content *and* receiver tables follows from the collusion resistance and frame-proofness of the instantiated fingerprinting scheme.

The mapping in Section 4.3 is an exact instantiation of the SSW scheme and therefore inherits its collusion resistance and frame-proofness (see Theorem 1). We note that the proof of Theorem 1, which appears in [15], covers both collusion resistance *and* frame-proofness, although the original text of the theorem only seems to cover collusion resistance. Collusion resistance, related to false negatives, is shown in [15, Section 3.4], whereas frame-proofness, related to false positives, is shown in [15, Section 3.2].

IND-CPA Security. We reduce the security of our Chameleon scheme to that of the PRSG with which it is instantiated. In order to prove IND-CPA security, we prove that the key stream produced by the Chameleon scheme is pseudo-random (see Definition 1). IND-CPA security of the proposed scheme follows by a simple reduction argument (see [28, Section 5.3.1]). To further strengthen the proof, we assume that the adversary is in possession of the master table and all receiver tables, although in practice the adversary only has one or several receiver tables.

By scaling the real values of the content coefficients to the integer domain (see Remark 1), we obtain a plaintext symbol space $\mathcal{P}$ with a cardinality Z defined by the content and the scaling factor ρ. In the remainder of this section we assume that the plaintext symbol space $\mathcal{P}$ and the key symbol space $\mathcal{K}$ are equal to $\{0, 1, \ldots, Z-1\}$. We make this assumption to simplify our notation, but stress that this is no restriction, as there is a one-to-one mapping between the actual plaintext symbol space $[0, z]$ and the scaled space $\{0, 1, \ldots, Z-1\}$, which enumerates the elements of $[0, z]$ starting from 0.[7] In the sequel, by *key symbols* we mean the elements of $\mathcal{K}$. We also note that the obvious choice for the group order p is the size of the symbol space: $p = |\mathcal{K}| = Z$. This ensures identical size of plaintext and ciphertext space.

The proof is divided into 4 major steps. First, we show the properties of the random variable that results from a single draw from the master table (Lemma 5). Second, we define these properties as the starting point of an iteration on the number s of draws from the master table (Definition 8). Third, we prove that the random variable that results from adding randomly drawn master table entries improves with every draw, where improving means being statistically closer to a truly random variable (Lemma 6). Last, we prove the pseudo-randomness of the Chameleon scheme's key stream (Theorem 3).

Lemma 5. *Let* $\mathsf{Pr}\left[X^{(1)} = x\right]$ *denote the probability of drawing the key symbol* $x \in \mathcal{K}$ *in a single draw from master table MT. Let* $\eta_k \in \{0, 1, \ldots, L\}$ *denote the number of times that key symbol* $x_k \in \mathcal{K}$ *appears in MT. When we select a master table entry at a random address with uniform distribution, then the probability of obtaining key symbol* $x_k \in \mathcal{K}$ *is* $p_k := \mathsf{Pr}\left[X^{(1)} = x_k\right] = \frac{\eta_k}{L}$.

[7] Note that $[0, z]$ consists of real numbers with finite precision. As pointed out in Remark 1 these real numbers are mapped to integers by applying a scaling factor ρ.

Proof. There are L entries in the master table. Due to the uniform distribution of the selected address, each master table entry has the same probability of being selected. Therefore, the probability of a specific key symbol $x_k \in \mathcal{K}$ being selected is the number η_k of occurrences of x_k in the master table divided by the total number L of master table entries.

For a single draw from the master table, the resulting random variable thus only depends on the number of occurrences of the key symbols within the master table. As the master table entries are generated with uniform distribution, the frequencies are unlikely to be identical for each key symbol, leading to a non-uniform and therefore insecure distribution $\mathsf{Pr}\left[X^{(1)}\right]$.

Definition 8 (Strong convergence). *Let U be a random variable uniformly distributed over the key symbol space. Let the* statistical quality *$SQ^{(1)}$ of MT be the statistical difference between $X^{(1)}$ and U: $SQ^{(1)} := \frac{1}{2}\sum_{k=0}^{Z-1}\left|p_k - \frac{1}{Z}\right|$. We call the master table* strongly converging *if $2SQ^{(1)} \leq d$ for some $d \in \mathbb{R}$ such that $d < 1$.*

The statistical quality $SQ^{(1)}$ is thus a measure for the initial suitability of the master table for generating a uniform distribution. The next lemma is the main result of the security analysis; it proves that the statistical quality $SQ^{(s)}$ gets better with every of the s draws.

Lemma 6. *Let U be a random variable uniformly distributed over the key symbol space. Let MT be a strongly converging master table. Let X_k denote the k-th draw from MT and $X^{(s)}$ the random variable resulting from s independent uniformly distributed draws added modulo Z: $X^{(s)} := \sum_{k=1}^{s} X_k \bmod Z$. Then the statistical difference $SQ^{(s)}$ between $X^{(s)}$ and U is a negligible function with an upper bound of $\frac{1}{2}d^s$.*

Proof. The proof is by induction. For all $k \in \mathcal{K}$, let $p_k^{(i)} := \mathsf{Pr}\left[X^{(i)} = k\right]$ denote the probability of the event that in the i-th iteration the random variable $X^{(i)}$ takes the value of key symbol k. Represent this probability with an additive error $e_k^{(i)}$ such that $p_k^{(i)} = \frac{1}{Z}(1 + e_k^{(i)})$. Due to $\sum_{k=0}^{Z-1} p_k^{(i)} = 1$, we obtain $\sum_{k=0}^{Z-1} e_k^{(i)} = 0$. The induction start is trivially fulfilled by every strongly converging master table: $SQ^{(1)} \leq \frac{1}{2}d$. As the induction hypothesis, we have $SQ^{(i)} \leq \frac{1}{2}d^i$, where $SQ^{(i)} := \frac{1}{2}\sum_{k=0}^{Z-1}|p_k^{(i)} - \frac{1}{Z}| = \frac{1}{2Z}\sum_{k=0}^{Z-1}|e_k^{(i)}|$. The induction claim is $SQ^{(i+1)} \leq \frac{1}{2}d^{i+1}$. The induction proof follows: Iteration $i+1$ is defined as $X^{(i+1)} := \sum_{k=1}^{i+1} X_k \bmod Z$, which is equal to $X^{(i+1)} = X^{(i)} + X_{i+1} \bmod Z$, where X_{i+1} is just a single draw with the probabilities p_k from Lemma 5 and error representation $p_k = \frac{1}{Z}(1 + e_k)$ such that $\sum_{k=0}^{Z-1} e_k = 0$. Therefore, we obtain for all $k \in \mathcal{K}$ that

$$\mathsf{Pr}\left[X^{(i+1)} = k\right] = \sum_{j=0}^{Z-1} \mathsf{Pr}\left[X^{(i)} = j\right] \cdot \mathsf{Pr}\left[X_{i+1} = (k-j) \bmod Z\right]$$

$$= \sum_{j=0}^{Z-1} p_j^{(i)} p_{(k-j) \bmod Z} = \frac{1}{Z^2} \sum_{j=0}^{Z-1} (1 + e_j^{(i)})(1 + e_{(k-j) \bmod Z})$$

$$= \frac{1}{Z^2} \left(\sum_{j=0}^{Z-1} 1 + \underbrace{\sum_{j=0}^{Z-1} e_j^{(i)}}_{=0} + \underbrace{\sum_{j=0}^{Z-1} e_{(k-j) \bmod Z}}_{=0} + \sum_{j=0}^{Z-1} e_j^{(i)} e_{(k-j) \bmod Z} \right)$$

$$= \frac{1}{Z} + \frac{1}{Z^2} \sum_{j=0}^{Z-1} e_j^{(i)} e_{(k-j) \bmod Z}$$

The upper bound for the statistical difference in iteration $i+1$ is

$$SQ^{(i+1)} := \frac{1}{2} \sum_{k=0}^{Z-1} \left| \mathsf{Pr}\left[X^{(i+1)} = k\right] - \frac{1}{Z} \right| = \frac{1}{2} \sum_{k=0}^{Z-1} \left| \frac{1}{Z^2} \sum_{j=0}^{Z-1} e_j^{(i)} e_{(k-j) \bmod Z} \right|$$

$$\leq \frac{1}{2Z^2} \left(\sum_{k=0}^{Z-1} \left| e_k^{(i)} \right| \right) \left(\sum_{k=0}^{Z-1} |e_k| \right) = 2SQ^{(i)} SQ^{(1)} \leq \frac{1}{2} d^{i+1},$$

where the first inequality follows from the fact that the two sums on the left-hand side run over every combination of $e_j^{(i)} e_{(k-j) \bmod Z}$, which may have opposite signs, whereas the right-hand side adds the absolute values of all combinations, avoiding any mutual elimination of combinations with opposite signs.

Note that the proof relies on the uniform sequence assumption, i.e., the addresses used to point into the master table have independent uniform distribution. Clearly, this assumption has to be slightly weakened in practice by replacing true randomness with pseudo-randomness. In Theorem 3 we therefore show that we can use pseudo-randomness without compromising security. The idea is to reduce an attack on the Chameleon key stream to an attack on the PRSG itself:

Theorem 3. *Let U be a random variable uniformly distributed over the key symbol space. Let MT be a strongly converging master table. Let the number $s(\lambda')$ of draws from MT be a polynomial function of the security parameter λ' of $\mathcal{CE}$ such that the statistical difference $SQ^{(s)}(\lambda')$ between $X^{(s)}$ and U is a negligible function under the uniform sequence assumption. Then even after replacement of the uniform sequence of addresses with a PRS, no probabilistic polynomial-time adversary can distinguish the pseudo-random key stream consisting of variables $X^{(s)}$ from a truly random key stream with variables U.*

Before we enter into the details of the proof, we clarify the attack goal, the adversary's capabilities, and the criteria for a successful break of (i) a PRSG and (ii) the pseudo-randomness of our Chameleon scheme's key stream:

(i) The goal of an adversary $\mathcal{A}$ attacking a PRSG is to distinguish the output of G on a random seed from a random string of identical length (see Definition 2).

$\mathcal{A}$'s capabilities are limited to a probabilistic Turing machine whose running time is polynomially bounded in the length of its input (and thus also in the security parameter λ, which is defines the input length). A successful break is defined as follows: The challenger $\mathcal{C}$ generates a random seed $str \xleftarrow{R} \{0,1\}^{\lambda}$ and a random string $str_1 \xleftarrow{R} \{0,1\}^{\mathsf{len}(\lambda)}$ with uniform distribution. $\mathcal{C}$ then applies the PRSG to str and obtains $str_0 \leftarrow \mathsf{G}(str)$. Finally, $\mathcal{C}$ tosses a coin $b \xleftarrow{R} \{0,1\}$ with uniform distribution and sends str_b to $\mathcal{A}$. The challenge for $\mathcal{A}$ is to distinguish the two cases, i.e., guess whether str_b was generated with the PRSG ($b = 0$) or the uniform distribution ($b = 1$). $\mathcal{A}$ wins if the guess b' is equal to b. The advantage of $\mathcal{A}$ is defined as:

$$\mathsf{Adv}\,(\lambda) := |\mathsf{Pr}\,[b' = 0 | b = 0] - \mathsf{Pr}\,[b' = 0 | b = 1]|\ , \tag{7}$$

where the randomness is taken over all coin tosses of $\mathcal{C}$ and $\mathcal{A}$.

(ii) The goal of adversary $\mathcal{A}$ attacking the pseudo-randomness of the Chameleon scheme's key stream is to distinguish n instances of $X^{(s)}$ from a truly random key stream. $\mathcal{A}$ is limited to a probabilistic Turing machine whose running time is polynomially bounded in the length of its input (and thus also in the security parameter λ', as this input is given in unary representation). A successful break is defined as follows: The challenger $\mathcal{C}$ generates a stream of n random keys: $K_1 := (k_{1,1}, \ldots, k_{1,n})$ such that $k_{1,j} \xleftarrow{R} \mathcal{K}$ for all $j \in \{1, \ldots, n\}$. Next, $\mathcal{C}$ generates a random seed $str \xleftarrow{R} \{0,1\}^{\lambda}$ and a strongly converging master table MT. Then $\mathcal{C}$ applies the PRSG to str in order to obtain a pseudo-random sequence of length $\mathsf{len}(\lambda) \geq n \cdot s \cdot l$, which is interpreted as a sequence of $n \cdot s$ addresses in the master table. Subsequently, $\mathcal{C}$ adds for each content coefficient m_j the corresponding s master table entries modulo Z to obtain the other key stream candidate: $K_0 := (k_{0,1}, \ldots, k_{0,n})$ such that $k_{0,j} \leftarrow \sum_{k=1}^{s} mt_{\beta_{j,k}} \bmod Z$. Finally, $\mathcal{C}$ tosses a coin $b \xleftarrow{R} \{0,1\}$ with uniform distribution and sends key stream candidate K_b to $\mathcal{A}$. The challenge for $\mathcal{A}$ is to distinguish the two cases, i.e., guess whether K_b was generated with the Chameleon scheme ($b = 0$) or the uniform distribution ($b = 1$). $\mathcal{A}$ wins if the guess b' is equal to b. The advantage is analogous to (7).

After definition of the attack games, we give the full proof of Theorem 3:

Proof. The proof is by contradiction. Assuming that the advantage of an adversary $\mathcal{A}$ against the pseudo-randomness of the Chameleon scheme's key stream is not negligible, we construct a distinguisher $\mathcal{A}'$ for the PRSG itself, contradicting the assumptions on the PRSG from Definition 2. We show the individual steps of constructing $\mathcal{A}'$ in Fig. 2.

1. The challenger $\mathcal{C}$ generates a random seed $str \xleftarrow{R} \{0,1\}^{\lambda}$ and a random string $str_1 \xleftarrow{R} \{0,1\}^{\mathsf{len}(\lambda)}$ with uniform distribution. $\mathcal{C}$ then applies the PRSG to str: $str_0 \leftarrow \mathsf{G}(str)$. Finally, $\mathcal{C}$ tosses a coin $b \xleftarrow{R} \{0,1\}$ with uniform distribution.
2. $\mathcal{C}$ sends str_b to $\mathcal{A}'$. $\mathcal{A}'$ needs to guess b.

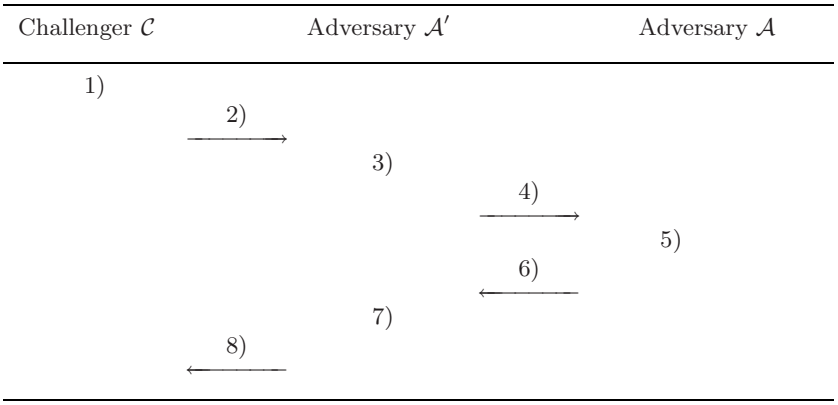


Fig. 2. Construction of adversary $\mathcal{A}'$ based on adversary $\mathcal{A}$

3. $\mathcal{A}'$ generates a strongly converging master table MT. Then $\mathcal{A}'$ takes the string str_b of length $\mathsf{len}(\lambda) \geq n \cdot s \cdot l$ and interprets it as a sequence of $n \cdot s$ addresses in the master table according to (3). Subsequently, $\mathcal{A}'$ adds for each content coefficient m_j the corresponding s master table entries modulo Z to obtain a key stream $K_b := (k_{b,1}, \ldots, k_{b,n})$ such that $k_{b,j} \leftarrow \sum_{k=1}^{s} mt_{\beta_{j,k}} \bmod Z$.
4. $\mathcal{A}'$ sends the key stream K_b to $\mathcal{A}$ as a challenge.
5. $\mathcal{A}$ calculates the guess b', where $b' = 0$ represents the random case, i.e., $\mathcal{A}$ guesses that K_b is a truly random key stream, and $b' = 1$ represents the pseudo-random case, i.e., $\mathcal{A}$ guesses that K_b was generated with the Chameleon scheme.
6. $\mathcal{A}$ sends the guess b' to $\mathcal{A}'$.
7. $\mathcal{A}'$ copies $\mathcal{A}$'s guess.
8. $\mathcal{A}'$ sends b' to $\mathcal{C}$ as a guess for b.

To finish the proof, we need to show that if the advantage of $\mathcal{A}$ against the pseudo-randomness of the Chameleon key stream is not negligible, then the advantage of $\mathcal{A}'$ against the PRSG is not negligible. We prove this by bounding the probability differences in the real attack scenario, where $\mathcal{A}$ is given input by a correct challenger, and the simulated attack, where $\mathcal{A}$ is given slightly incorrect input by $\mathcal{A}'$. The contradictive assumption is that $\mathcal{A}$'s advantage against the Chameleon encryption scheme is not negligible in the real attack:

$$\left|\mathsf{Pr}^{\mathsf{real}}\left[b' = 0 | b = 0\right] - \mathsf{Pr}^{\mathsf{real}}\left[b' = 0 | b = 1\right]\right| \geq \epsilon_{\mathsf{CE}}(\lambda') \;,$$

where $\mathsf{Pr}^{\mathsf{real}}[\,]$ denotes probabilities in the real attack between a Chameleon challenger and a Chameleon adversary $\mathcal{A}$ and $\epsilon_{\mathsf{CE}}(\lambda')$ is $\mathcal{A}$'s advantage, which is not negligible. The randomness is taken over all coin tosses of $\mathcal{C}$ and $\mathcal{A}$.

Next, we summarize the input to $\mathcal{A}$ in the real attack and the simulated attack. In the real attack, $\mathcal{A}$ obtains either the key stream output K_0 of the Chameleon scheme on a truly random seed str ($b = 0$), or a truly random key stream K_1 ($b = 1$). Specifically, the key stream element $k_{0,j}$ of K_0 is equal to $k_{0,j} = \sum_{k=1}^{s} mt_{\beta_{j,k}} \bmod Z$, where the truly random seed str determines the addresses of the master table entries mt_j via the PRSG according to (3).

In the simulated attack, $\mathcal{A}'$ does not apply the PRSG and instead uses the challenge str_b as a shortcut. $\mathcal{A}$ obtains either the key stream output K_0 of the Chameleon scheme executed on a pseudo-random string str_0, derived from a truly random seed str ($b = 0$), or the key stream output K_1 of the Chameleon scheme executed on a truly random string str_1 ($b = 1$). The key stream outputs K_0 and K_1 in the simulated attack thus only differ by the fact that K_0 comes from a pseudo-random string and K_1 from a truly random string.

There is no difference between real and simulated attack for $b = 0$. The key stream outputs K_0^{real} and K_0^{sim} both come from a PRSG executed on a truly random seed str, leading to the following relation:

$$\left|\mathsf{Pr}^{\mathsf{real}}\left[b' = 0 | b = 0\right] - \mathsf{Pr}^{\mathsf{sim}}\left[b' = 0 | b = 0\right]\right| = 0 \ ,$$

where the randomness is taken over all coin tosses of $\mathcal{C}$ and $\mathcal{A}$ in the real attack and those of $\mathcal{C}$, $\mathcal{A}'$ and $\mathcal{A}$ in the simulated attack.

For $b = 1$ and a real attack, $\mathcal{A}$ obtains a truly random key stream K_1^{real}. In the simulated attack, $\mathcal{A}'$ operates on a truly random string str_1 that determines $n \cdot s$ addresses according to (3). As str_1 is truly random, the $n \cdot s$ addresses are also truly random with independent uniform distribution. Combined with the assumptions of the theorem, this implies that each pair of key stream elements in real and simulated attack has a negligible statistical difference. Negligible statistical difference implies polynomial-time indistinguishability [24, Section 3.2.2]. Let $\epsilon_{\mathsf{diff}}(\lambda')$ be the corresponding negligible bound on the advantage of a distinguisher, which applies for one key stream element. Then the difference between both attacks for all n key stream elements has a negligible upper bound $n \cdot \epsilon_{\mathsf{diff}}(\lambda')$:

$$\left|\mathsf{Pr}^{\mathsf{real}}\left[b' = 0 | b = 1\right] - \mathsf{Pr}^{\mathsf{sim}}\left[b' = 0 | b = 1\right]\right| \leq n \cdot \epsilon_{\mathsf{diff}}(\lambda') \ ,$$

where the randomness is taken over all coin tosses of $\mathcal{C}$ and $\mathcal{A}$ in the real attack and those of $\mathcal{C}$, $\mathcal{A}'$ and $\mathcal{A}$ in the simulated attack.

The last three inequalities lead to a lower bound for the success probability of $\mathcal{A}$ in the simulated attack, which is also the success probability of $\mathcal{A}'$ in the attack against the PRSG:

$$\left|\mathsf{Pr}^{\mathsf{sim}}\left[b' = 0 | b = 0\right] - \mathsf{Pr}^{\mathsf{sim}}\left[b' = 0 | b = 1\right]\right| \geq \epsilon_{\mathsf{CE}}(\lambda') - n \cdot \epsilon_{\mathsf{diff}}(\lambda')$$

As $\epsilon_{\mathsf{CE}}(\lambda')$ is not negligible by the contradictive assumption, $\epsilon_{\mathsf{diff}}(\lambda')$ is negligible by the negligible statistical difference and n is a constant, we conclude that the

success probability of $\mathcal{A}'$ against the PRSG is not negligible, completing the contradiction and the proof.

5 Implementation

The master table MT obviously becomes strongly converging for sufficiently large L. Our simulation shows that $L = 4Z$ gives high assurance of strong convergence. However, lower values still lead to weak convergence in the sense that it is not proven by our upper bound, but can easily be verified numerically. As discussed in Section 4.2 we need to choose the number s of draws from MT in accordance with L. The upper bound in Theorem 6 is too conservative to choose s in practice. Our simulation shows that the statistical difference $SQ^{(s)}$ not only decreases with factor $d \approx 2SQ^{(1)} < 1$, but with an even smaller factor. This is due to the fact that some of the combinations $e_j^{(i)} e_{(k-j) \bmod Z}$ on the left-hand side of the inequality in the proof of Lemma 6 cancel out. In Appendix F we therefore give an explicit formula for calculation of the *exact* statistical difference after s draws from MT. The center can thus generate MT with arbitrary length L, numerically verify convergence and determine the minimum number of draws s_{min} that provides the desired statistical difference.

The content representation can be extended to cover movies and songs by interpreting them as a sequence of content items. A straightforward approach is to regularly refresh the session key. While further refinements are possible, aiming to prevent sequence-specific attacks such as averaging across movie frames, they are beyond the scope of this document. However, it remains to define how the insignificant part of the content should be processed (see Section 3.2). There are three obvious options: sending it in the clear, passing it through our scheme or encrypting it separately. Note that by its very definition, this part does not give significant information about the content and was not watermarked because the coefficients do not have perceptible influence on the reassembled content. The easiest option is thus to pass them through the proposed scheme, which does not influence goodness and maintains confidentiality of the content.

At first sight our proposed scheme trivially fulfills the correctness requirement (see Definition 4) due to the correctness of the SSW scheme. However, both schemes face difficulties in the rare event that a content coefficient is at the lower or upper end of the interval $[0, z]$, which corresponds with plaintext symbols close to 0 or $Z - 1$. If the additive fingerprint coefficient causes a trespass of lower or upper bound, the SSW scheme needs to decrease the coefficient's amplitude and round to the corresponding bound. Similarly, our scheme must avoid a wrap-around in the additive group, e.g., when plaintext symbol $Z - 2$ obtains a coefficient of $+3$ and ends up at 1 after decryption. There are many options with different security trade-offs, such as sending a flag or even sending the coefficient in cleartext; the appropriate choice depends on further requirements of the implementation. Note that the center trivially anticipates the occurrence of a wrap-around from inspecting the content coefficients.

5.1 Efficiency

Three performance parameters determine whether the proposed scheme is efficient and implementable: transmission overhead, storage size of a receiver, and computation cost. We stress that our scheme enables a tradeoff between storage size and computation cost. Increasing the size L of the master table (and thus the storage size) decreases the necessary number s of draws (and thus the computation cost), as can be seen from Lemma 6 and Definition 8, where $SQ^{(1)}$ and thus d decreases with L. This feature allows us to adapt the scheme to the particular constraints of the receiver, in particular to decrease s.

The transmission overhead of the Chameleon scheme is 0 if the master table and receiver tables are not renewed on a regular basis. In this scenario, the Chameleon scheme's transmission overhead is 0 because ciphertext and cleartext use the same symbol space and thus have the same length; the transmission overhead of fingercasting is thus determined by that of the broadcast encryption scheme, which is moderate [5,6,7,8].[8]

For the storage size, we highlight the parameters of a computation-intensive implementation. Let the content be an image with $n = 10,000$ significant coefficients of 16 bit length, such that $Z = 2^{16}$. By testing several lengths L of the master table MT, we found a statistical quality of $SQ^{(1)} = d/2 < 1/8$ for $L = 8 \cdot Z = 8 \cdot 2^{16} = 2^{19} = 2^l$. A receiver table thus has $2^{19} \cdot 16 = 2^{23}$ bit or 2^{20} Byte $= 2^{10}$ kByte $= 1$ MByte, which seems acceptable in practice.

The computation cost depends mostly on the number s of draws from the master table. To achieve a small statistical difference $SQ^{(s)}$, e.g., below 2^{-128}, we choose $s = 64$ and therefore $SQ^{(s)} < 1/2 \cdot d^s = 2^{-1} \cdot 2^{-2 \cdot 64} = 2^{-129}$ by the conservative upper bound of Lemma 6. Compared to a conventional stream cipher that encrypts $n \cdot \log_2 Z$ bits, a receiver has to generate $n \cdot s \cdot l$ pseudo-random bits, which is an overhead of $(s \cdot l)/\log_2 Z = 76$. To generate the pseudo-random key stream, the receiver has to perform $n \cdot s$ table lookups and $n \cdot (s+1)$ modular operations in a group of size 2^{16}.

In further tests, we also found a more storage-intensive implementation with $L = 2^{25}$ and $s = 25$, which leads to 64 MBytes of storage and an overhead of $(s \cdot l)/\log_2 Z \approx 39$. By calculating the exact statistical difference of Appendix F instead of the conservative upper bound of Lemma 6, s decreases further, but we are currently unaware of any direct formula to calculate s based on a master table length L and a desired statistical difference $SQ^{(s)}$ (or vice versa).

If the security requirements of an implementation require a regular renewal of the master table and the subsequent redistribution of the receiver tables, then the transmission overhead obviously increases. For each redistribution, the total key material to be transmitted has the size of the master table times the number of receivers. As mentioned before, a redistribution channel then becomes necessary if the broadcast channel does not have enough spare bandwidth.

[8] For example, this overhead is far smaller than that of the trivial solution, which consists of sequentially sending an individually fingerprinted copy of the content individually encrypted over the broadcast channel.

6 Conclusion and Open Problems

In this document we gave a formal proof of the security of a new Chameleon cipher. Applied to a generic fingercasting approach, it provides confidentiality of ciphertext, traceability of content and keys as well as renewability. We achieved confidentiality through a combination of a generic broadcast encryption (BE) scheme and the new Chameleon cipher. The BE scheme provides a fresh session key, which the Chameleon scheme uses to generate a pseudo-random key stream. The pseudo-random key stream arises from adding key symbols at pseudo-random addresses in a long master table, initially filled with random key symbols. We have reduced the security of the pseudo-random key stream to that of a pseudo-random sequence generator.

In addition, we achieved traceability of keys and content through embedding of a receiver-specific fingerprint into the master table copies, which are given to the receivers. During decryption, these fingerprints are inevitably embedded into the content, enabling the tracing of malicious users. We achieve the same collusion resistance as an exemplary watermarking scheme with proven security bound. It may be replaced with any fingerprinting scheme whose watermarks can be decomposed into additive components. Finally, we achieved renewability through revocation, which is performed in the BE scheme.

Two open problems are the most promising for future work. First of all, the detection algorithm should be extended in order to allow blind detection of a watermark even in the absence of the original content. Another open problem is to combine Chameleon encryption with a code-based fingerprinting scheme in the sense of Boneh and Shaw [29]. The master table in Chameleon would need to embed components of codewords in such a way that a codeword is embedded into the content.

References

1. Adelsbach, A., Huber, U., Sadeghi, A.R.: Fingercasting—joint fingerprinting and decryption of broadcast messages. Tenth Australasian Conference on Information Security and Privacy—ACISP 2006, Melbourne, Australia, July 3-5, 2006. Volume 4058 of Lecture Notes in Computer Science, Springer (2006)
2. Touretzky, D.S.: Gallery of CSS descramblers. Webpage, Computer Science Department of Carnegie Mellon University (2000) URL `http://www.cs.cmu.edu/~dst/DeCSS/Gallery` (November 17, 2005).
3. 4C Entity, LLC: CPPM specification—introduction and common cryptographic elements. Specification Revision 1.0 (2003) URL `http://www.4centity.com/data/tech/spec/cppm-base100.pdf`.
4. AACS Licensing Administrator: Advanced access content system (AACS): Introduction and common cryptographic elements. Specification Revision 0.90 (2005) URL `http://www.aacsla.com/specifications/AACS_Spec-Common_0.90.pdf`.
5. Fiat, A., Naor, M.: Broadcast encryption. In Stinson, D.R., ed.: CRYPTO 1993. Volume 773 of Lecture Notes in Computer Science, Springer (1994) 480–491
6. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In Kilian, J., ed.: CRYPTO 2001. Volume 2139 of Lecture Notes in Computer Science, Springer (2001) 41–62

7. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In Yung, M., ed.: CRYPTO 2002. Volume 2442 of Lecture Notes in Computer Science, Springer (2002) 47–60
8. Jho, N.S., Hwang, J.Y., Cheon, J.H., Kim, M.H., Lee, D.H., Yoo, E.S.: One-way chain based broadcast encryption schemes. In Cramer, R., ed.: EUROCRYPT 2005. Volume 3494 of Lecture Notes in Computer Science, Springer (2005) 559–574
9. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In Desmedt, Y., ed.: CRYPTO 1994. Volume 839 of Lecture Notes in Computer Science, Springer (1994) 257–270
10. Naor, M., Pinkas, B.: Threshold traitor tracing. In Krawczyk, H., ed.: CRYPTO 1998. Volume 1462 of Lecture Notes in Computer Science, Springer (1998) 502–517
11. Kundur, D., Karthik, K.: Video fingerprinting and encryption principles for digital rights management. Proceedings of the IEEE **92**(6) (2004) 918–932
12. Anderson, R.J., Manifavas, C.: Chameleon—a new kind of stream cipher. In Biham, E., ed.: FSE 1997. Volume 1267 of Lecture Notes in Computer Science, Springer (1997) 107–113
13. Briscoe, B., Fairman, I.: Nark: Receiver-based multicast non-repudiation and key management. In: ACM EC 1999, ACM Press (1999) 22–30
14. Cox, I.J., Kilian, J., Leighton, T., Shamoon, T.: Secure spread spectrum watermarking for multimedia. IEEE Transactions on Image Processing **6**(12) (1997) 1673–1687
15. Kilian, J., Leighton, F.T., Matheson, L.R., Shamoon, T.G., Tarjan, R.E., Zane, F.: Resistance of digital watermarks to collusive attacks. Technical Report TR-585-98, Princeton University, Department of Computer Science (1998) URL: `ftp://ftp.cs.princeton.edu/techreports/1998/585.ps.gz`.
16. Anderson, R.J., Kuhn, M.: Tamper resistance—a cautionary note. In Tygar, D., ed.: USENIX Electronic Commerce 1996, USENIX (1996) 1–11
17. Maurer, U.M.: A provably-secure strongly-randomized cipher. In Damgård, I., ed.: EUROCRYPT 1990. Volume 473 of Lecture Notes in Computer Science, Springer (1990) 361–373
18. Maurer, U.: Conditionally-perfect secrecy and a provably-secure randomized cipher. Journal of Cryptology **5**(1) (1992) 53–66
19. Ferguson, N., Schneier, B., Wagner, D.: Security weaknesses in a randomized stream cipher. In Dawson, E., Clark, A., Boyd, C., eds.: ACISP 2000. Volume 1841 of Lecture Notes in Computer Science, Springer (2000) 234–241
20. Ergün, F., Kilian, J., Kumar, R.: A note on the limits of collusion-resistant watermarks. In Stern, J., ed.: EUROCRYPT 1999. Volume 1592 of Lecture Notes in Computer Science, Springer (1999) 140–149
21. Brown, I., Perkins, C., Crowcroft, J.: Watercasting: Distributed watermarking of multicast media. In Rizzo, L., Fdida, S., eds.: Networked Group Communication 1999. Volume 1736 of Lecture Notes in Computer Science, Springer (1999) 286–300
22. Parviainen, R., Parnes, P.: Large scale distributed watermarking of multicast media through encryption. In Steinmetz, R., Dittmann, J., Steinebach, M., eds.: Communications and Multimedia Security (CMS 2001). Volume 192 of IFIP Conference Proceedings., International Federation for Information Processing, Communications and Multimedia Security (IFIP), Kluwer (2001) 149–158
23. Luh, W., Kundur, D.: New paradigms for effective multicasting and fingerprinting of entertainment media. IEEE Communications Magazine **43**(5) (2005) 77–84
24. Goldreich, O.: Basic Tools. First edn. Volume 1 of Foundations of Cryptography. Cambridge University Press, Cambridge, UK (2001)

25. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Okamoto, T., ed.: ASIACRYPT 2000. Volume 1976 of Lecture Notes in Computer Science, Springer (2000) 531–545
26. National Institute of Standards and Technology, *Announcing the Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication FIPS PUB 197, November 26, 2001, URL `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf` .
27. National Institute of Standards and Technology, *Data Encryption Standard (DES)*, Federal Information Processing Standards Publication FIPS PUB 46-3, October 25, 1999, URL `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf` .
28. Goldreich, O.: Basic Applications. First edn. Volume 2 of Foundations of Cryptography. Cambridge University Press, Cambridge, UK (2004)
29. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data (extended abstract). In Coppersmith, D., ed.: CRYPTO 1995. Volume 963 of Lecture Notes in Computer Science, Springer (1995) 452–465

A Abbreviations

Table 1 summarizes all abbreviations used in this document.

Table 1. Abbreviations used in this document

Abbreviation	Abbreviated Technical Term
AACS	Advanced Access Content System
AES	Advanced Encryption Standard
BE	Broadcast Encryption
CPPM	Content Protection for Pre-Recorded Media
CRL	Certificate Revocation List
CSS	Content Scrambling System
DCT	Discrete Cosine Transform
DES	Data Encryption Standard
DVD	Digital Versatile Disc
FE	Fingerprint Embedding
PRS	Pseudo-Random Sequence
PRSG	Pseudo-Random Sequence Generator
SSW	Spread Spectrum Watermarking
TV	Television

B Summary of Relevant Parameters

Table 2 summarizes all parameters of our fingercasting approach and the underlying fingerprinting scheme, which we instantiate with the SSW scheme of [15].

Table 2. Parameters of the proposed fingercasting scheme and the SSW scheme

Parameter	Description
N	Number of receivers
u_i	i-th receiver
q	Maximum tolerable number of colluding receivers
M	Representation of the original content
m_j	j-th coefficient of content M
n	Number of coefficients (Chameleon scheme)
n'	Number of coefficients (fingerprinting scheme)
$CF^{(i)}$	Content fingerprint of receiver u_i
$cf_j^{(i)}$	Coefficient j of u_i's content fingerprint $CF^{(i)}$
M^*	Illegal copy of the original content
CF^*	Fingerprint found in an illegal copy M^*
C	Ciphertext of the original content M
c_j	j-th coefficient of ciphertext C
k^{sess}	Session key used as a seed for the PRSG
MT	Master table of the Chameleon scheme
α	Address of a table entry
mt_α	α-th entry of the master table MT
$TF^{(i)}$	Table fingerprint for receiver table of receiver u_i
$tf_\alpha^{(i)}$	h-th coefficient of u_i's table fingerprint $TF^{(i)}$
$RT^{(i)}$	Receiver table of receiver u_i
$rt_\alpha^{(i)}$	α-th entry of the receiver table $RT^{(i)}$
l	Number of bits needed for the binary address of a table entry
L	Number of entries of the tables, $L = 2^l$
F	Number of fingerprinted entries of a receiver table
s	Number of master table entries per ciphertext coefficient
par_{CE}	Input parameters (Chameleon scheme)
par_{FP}	Input parameters (fingerprinting scheme)
σ	Standard deviation for receiver table
σ'	Standard deviation for SSW scheme
p^{bad}	Maximum probability of a bad copy
p^{pos}	Maximum probability of a false positive
p^{neg}	Maximum probability of a false negative
δ	Goodness criterion (SSW scheme)
t	Threshold of similarity measure (SSW scheme)
dec	Decision output of detection algorithm
z	Upper bound of interval $[0, z]$ (content coefficients)
Z	Key space size and cardinality of discrete interval $[0, z]$
ρ	Scaling factor from real numbers to group elements
p	Order of the additive group

C Chameleon Encryption

Definition 9. *A* Chameleon encryption scheme *is a tuple of five polynomial-time algorithms* $\mathcal{CE} := (\mathsf{KeyGenCE}, \mathsf{KeyExtrCE}, \mathsf{EncCE}, \mathsf{DecCE}, \mathsf{DetectCE})$*, where:*

- $\mathsf{KeyGenCE}$ *is the probabilistic* key generation *algorithm used by the center to set up all parameters of the scheme.* $\mathsf{KeyGenCE}$ *takes the number* N *of receivers, a security parameter* λ'*, and a set of performance parameters* par_{CE} *as input in order to generate a secret master table* MT*, a tuple* $TF := (TF^{(1)}, \ldots, TF^{(N)})$ *of secret table fingerprints containing one fingerprint per receiver, and a threshold* t*. The values* N *and* λ' *are public:*

$$(MT, TF, t) \leftarrow \mathsf{KeyGenCE}(N, 1^{\lambda'}, par_{\mathsf{CE}})$$

- $\mathsf{KeyExtrCE}$ *is the deterministic* key extraction *algorithm used by the center to extract the secret receiver table* $RT^{(i)}$ *to be delivered to receiver* u_i *in the setup phase.* $\mathsf{KeyExtrCE}$ *takes the master table* MT*, the table fingerprints* TF*, and the index* i *of receiver* u_i *as input in order to return* $RT^{(i)}$*:*

$$RT^{(i)} \leftarrow \mathsf{KeyExtrCE}(MT, TF, i)$$

- EncCE *is the deterministic* encryption *algorithm used by the center to encrypt content* M *such that only receivers in possession of a receiver table and the session key can recover it.* EncCE *takes the master table* MT*, a session key* k^{sess}*, and content* M *as input in order to return the ciphertext* C*:*

$$C \leftarrow \mathsf{EncCE}(MT, k^{\mathsf{sess}}, M)$$

- DecCE *is the deterministic* decryption *algorithm used by a receiver* u_i *to decrypt a ciphertext* C*.* DecCE *takes the receiver table* $RT^{(i)}$ *of receiver* u_i*, a session key* k^{sess}*, and a ciphertext* C *as input. It returns a good copy* $M^{(i)}$ *of the underlying content* M *if* C *is a valid encryption of* M *using* k^{sess}*:*

$$M^{(i)} \leftarrow \mathsf{DecCE}(RT^{(i)}, k^{\mathsf{sess}}, C)$$

- $\mathsf{DetectCE}$ *is the deterministic fingerprint detection algorithm used by the center to detect whether the table fingerprint* $TF^{(i)}$ *of receiver* u_i *left traces in an illegal copy* M^**.* $\mathsf{DetectCE}$ *takes the original content* M*, the illegal copy* M^**, the session key* k^{sess}*, the table fingerprint* $TF^{(i)}$ *of* u_i*, and the threshold* t *as input in order to return* $dec = \texttt{true}$ *if the similarity measure of the underlying fingerprinting scheme indicates that the similarity between* M^* *and* $M^{(i)}$ *is above the threshold* t*. Otherwise it returns* $dec = \texttt{false}$*:*

$$dec \leftarrow \mathsf{DetectCE}(M, M^*, k^{\mathsf{sess}}, TF^{(i)}, t)$$

Correctness of $\mathcal{CE}$ *requires that*

$$\forall u_i \in \mathcal{U}: \ \mathsf{DecCE}(RT^{(i)}, k^{\mathsf{sess}}, \mathsf{EncCE}(MT, k^{\mathsf{sess}}, M)) = M^{(i)} \quad \text{such that}$$
$$\mathsf{Good}(M^{(i)}, M) = \texttt{true} \quad \text{(see Definition 3) with high probability.}$$

D Fingerprinting and Spread Spectrum Watermarking

In this section, we detail our notation of a fingerprinting scheme by describing the respective algorithms of Spread Spectrum Watermarking [14,15]. This scheme is a tuple of three polynomial-time algorithms (SetupFP, EmbedFP, DetectFP). We detail each of the three algorithms in Sections D.1–D.3.

D.1 Setup Algorithm

SetupFP is the probabilistic setup algorithm used by the center to set up all parameters of the scheme. SetupFP takes the number N of receivers, the number n' of content coefficients, a goodness criterion δ, a maximum probability p^{bad} of bad copies, and a maximum probability p^{pos} of false positives as input in order to return a tuple of secret content fingerprints CF, containing one fingerprint per receiver, as well as a similarity threshold t. The values N and n' are public:

$$(CF, t) \leftarrow \mathsf{SetupFP}(N, n', \delta, p^{\mathsf{bad}}, p^{\mathsf{pos}})$$

The algorithm of [14,15] proceeds as follows. The set of content fingerprints CF is defined as $CF := (CF^{(1)}, \ldots, CF^{(N)})$. The content fingerprint $CF^{(i)}$ of receiver u_i is a vector $CF^{(i)} := (cf_1^{(i)}, \ldots, cf_{n'}^{(i)})$ of n' fingerprint coefficients. For each receiver index $i \in \{1, \ldots, N\}$ and for each coefficient index $j \in \{1, \ldots, n'\}$, the fingerprint coefficient follows an independent normal distribution. The standard deviation of this distribution depends on the values N, n', δ, and p^{bad}:

$$\forall 1 \leq i \leq N, \forall 1 \leq j \leq n' : \quad cf_j^{(i)} \leftarrow \mathsf{N}(0, \sigma') \quad \text{with} \quad \sigma' = \mathsf{f}_{\sigma'}(N, n', \delta, p^{\mathsf{bad}})$$

The similarity threshold t is a function $t = \mathsf{f}_t(\sigma', N, p^{\mathsf{pos}})$ of σ', N, and p^{pos}. The details of $\mathsf{f}_{\sigma'}$ and f_t can be found in [15].

D.2 Watermark Embedding Algorithm

EmbedFP is the deterministic watermark embedding algorithm used by the center to embed the content fingerprint $CF^{(i)}$ of receiver u_i into the original content M. EmbedFP takes the original content M and the secret content fingerprint $CF^{(i)}$ of receiver u_i as input in order to return the fingerprinted copy $M^{(i)}$ of u_i:

$$M^{(i)} \leftarrow \mathsf{EmbedFP}(M, CF^{(i)})$$

The algorithm of [14,15] adds the fingerprint coefficient to the original content coefficient to obtain the fingerprinted content coefficient:

$$\forall j \in \{1, \ldots, n'\} : \quad m_j^{(i)} \leftarrow m_j + cf_j^{(i)}$$

D.3 Watermark Detection Algorithm

DetectFP is the deterministic watermark detection algorithm used by the center to verify whether an illegal content copy M^* contains traces of the content

fingerprint $CF^{(i)}$ that was embedded into the content copy $M^{(i)}$ of receiver u_i. DetectFP takes the original content M, the illegal copy M^*, the content fingerprint $CF^{(i)}$, and the similarity threshold t as input and returns the decision $dec \in \{\texttt{true}, \texttt{false}\}$:

$$dec \leftarrow \mathsf{DetectFP}(M, M^*, CF^{(i)}, t)$$

The algorithm of [14,15] calculates the similarity measure between the fingerprint in the illegal copy and the fingerprint of the suspect receiver. The similarity measure is defined as the dot product between the two fingerprints, divided by the Euclidean norm of the fingerprint in the illegal copy:

$$\begin{aligned} CF^* &\leftarrow M^* - M \\ \mathsf{Sim}(CF^*, CF^{(i)}) &\leftarrow \frac{CF^* \cdot CF^{(i)}}{||CF^*||} \\ \textbf{If } \mathsf{Sim}(CF^*, CF^{(i)}) &> t \\ &\textbf{Then Return } dec = \texttt{true} \\ &\textbf{Else Return } dec = \texttt{false} \end{aligned}$$

E Broadcast Encryption

In this section we describe a general BE scheme that allows revocation of an arbitrary subset of the set of receivers. Examples for such BE schemes are [6,7,8]. As these schemes all belong to the family of subset cover schemes defined in [6], we use this name to refer to them:

Definition 10. *A* Subset Cover BE (SCBE) scheme *is a tuple of four polynomial-time algorithms (*KeyGenBE*,* KeyExtrBE*,* EncBE*,* DecBE*), where:*

- KeyGenBE *is the probabilistic* key generation *algorithm used by the center to set up all parameters of the scheme.* KeyGenBE *takes the number N of receivers and a security parameter λ'' as input in order to generate the secret master key MK. The values N and λ'' are public:*

$$MK \leftarrow \mathsf{KeyGenBE}(N, 1^{\lambda''})$$

- KeyExtrBE *is the deterministic* key extraction *algorithm used by the center to extract the secret key $SK^{(i)}$ to be delivered to a receiver u_i in the setup phase.* KeyExtrBE *takes the master key MK and the receiver index i as input in order to return the secret key $SK^{(i)}$ of u_i:*

$$SK^{(i)} \leftarrow \mathsf{KeyExtrBE}(MK, i)$$

- EncBE *is the deterministic* encryption *algorithm used to encrypt session key k^{sess} in such a way that only the non-revoked receivers can recover it.* EncBE *takes the master key MK, the set $\mathcal{R}$ of revoked receivers, and session key k^{sess} as input in order to return the ciphertext C_{BE}:*

$$C_{\mathsf{BE}} \leftarrow \mathsf{EncBE}(MK, \mathcal{R}, k^{\mathsf{sess}})$$

– DecBE *is the deterministic* decryption *algorithm used by a receiver* u_i *to decrypt a ciphertext* C_{BE}. DecBE *takes the index* i *of* u_i*, its private key* $SK^{(i)}$*, and a ciphertext* C_{BE} *as input in order to return the session key* k^{sess} *if* C_{BE} *is a valid encryption of* k^{sess} *and* u_i *is non-revoked, i.e.,* $u_i \notin \mathcal{R}$*. Otherwise, it returns the failure symbol* $\perp$*:*

$$k^{\mathsf{sess}} \leftarrow \mathsf{DecBE}(i, SK^{(i)}, C_{\mathsf{BE}}) \quad \textit{if} \quad u_i \notin \mathcal{R}$$

Correctness of a SCBE scheme requires that

$$\forall u_i \in \mathcal{U} \setminus \mathcal{R}: \quad \mathsf{DecBE}(i, SK^{(i)}, \mathsf{EncBE}(MK, \mathcal{R}, k^{\mathsf{sess}})) = k^{\mathsf{sess}}\,.$$

F Selection of the Minimum Number of Draws

The center can calculate the statistical difference after s draws if it knows the corresponding probability distribution. The next lemma gives an explicit formula for this probability distribution. To determine the minimum number of draws to achieve a maximum statistical difference, e.g., 2^{-128}, the center increases s until the statistical difference is below the desired maximum. Note that this only needs to be done once at setup time of the system when s is chosen.

Lemma 7. *If the draws use addresses with independent uniform distribution and the master table MT is given in the representation of Lemma 5, then the drawing and adding of s master table entries leads to the random variable*

$$\begin{aligned}
X^{(s)} &:= \left(\sum_{j=1}^{s} X_j\right) \bmod Z \quad \textit{with} \\
\mathsf{Pr}\left[X^{(s)} = x\right] &= \sum_{\text{condition}} \binom{s}{s_0, \ldots, s_{Z-1}} \prod_{k=0}^{Z-1} {p_k}^{s_k} \\
\textit{where condition} &\Leftrightarrow (8) \wedge (9) \wedge (10): \\
s_k &\geq 0 \quad \forall\, k \in \{0, 1, \ldots, Z-1\} && (8) \\
\sum_{k=0}^{Z-1} s_k &= s && (9) \\
\left(\sum_{k=0}^{Z-1} s_k \cdot x_k\right) \bmod Z &= x\,, && (10)
\end{aligned}$$

where s_k *denotes the number of times that key space element* x_k *was chosen in the* s *selections and* $\binom{s}{s_0,\ldots,s_{Z-1}} := \frac{s!}{s_0! \cdot \ldots \cdot s_{Z-1}!}$ *denotes the multinomial coefficient.*

Proof. Each of the s selections is a random variable X_j with $\mathsf{Pr}\left[X_j = x_k\right] = p_k$. The independence of the random addresses transfers to the independence of the X_j. The probability of a complete set of s selections is thus a product of s probabilities of the form $\prod_1^s p$ with appropriate indices. The counter s_k stores

the number of times that probability p_k appears in this term. This counter is non-negative, implying(8). In total, there are s selections, implying (9).

To fulfill the condition $X^{(s)} = x$, the addition modulo Z of the s random variables must have the result x. Given the counters s_k, the result of the addition is $(\sum_{k=0}^{Z-1} s_k \cdot x_k) \bmod Z$. The combination of both statements implies (10).

There is more than one possibility for selecting s_k times the key symbol x_k during the s selections. Considering all such key symbols in s selections, the total number of possibilities is the number of ways in which we can choose s_0 times the key symbol x_0, then s_1 times the key symbol x_1, and so forth until we reach a total of s selections. This number is the multinomial coefficient $\binom{s}{s_0,\ldots,s_{Z-1}}$.

Note that we can trivially verify that the probabilities of all key space elements x in Lemma 7 add to 1. Among the three conditions (8), (9), and (10), the first two conditions appear in the well-known multinomial theorem

$$(\sum_{k=0}^{Z-1} p_k)^s = \sum_{\substack{s_0,\ldots,s_{Z-1}\geq 0 \\ s_0+\ldots+s_{Z-1}=s}} \binom{s}{s_0,\ldots,s_{Z-1}} \prod_{k=0}^{Z-1} {p_k}^{s_k}$$

By adding the probabilities over all elements, we obviously add over all addends on the right-hand side of the multinomial theorem. As the left-hand side trivially adds to 1, so do the probabilities over all key space elements.