VisAuth: Authentication over a Visual Channel using an Embedded Image

Jack Sturgess (✉) and Ivan Martinovic

Department of Computer Science, University of Oxford, Oxford, UK
{firstname.surname}@cs.ox.ac.uk

Abstract. Mobile payment systems are pervasive; their design is driven by convenience and security. In this paper, we identify five common problems in existing systems: (i) specialist hardware requirements, (ii) no reader-to-user authentication, (iii) use of invisible channels, (iv) dependence on a client-server connection, and (v) no inherent fraud detection. We then propose a novel system which overcomes these problems, so as to mutually authenticate a user, a point-of-sale reader, and a verifier over a visual channel, using an embedded image token to transport information, while providing inherent unauthorised usage detection. We show our system to be resilient against replay and tampering attacks.

1 Introduction

The popularity of cashless payments has risen sharply in recent years, surpassing cash payments in some places [1]. Consumers moved from cash to cashless payment cards primarily for convenience, not security—the first generation of magnetic strip payment cards were authenticated with an easily-forged, handwritten signature. These systems were widely replaced with Europay, MasterCard, and Visa (EMV)[1] payment card systems, protected by a chip and secret personal identification number (PIN). The payment card is inserted into a specialist reader and the PIN is entered and verified by the chip to authenticate the user and authorise the payment; more recent cards support contactless payments using near field communication (NFC) between card and reader.

Attacks (*e.g.*, [2, 3]) on payment card systems and incidences of fraud continue to occur, so consumers move to new cashless systems with the promise of greater security and convenience. Furthermore, carrying a dedicated payment card is increasingly regarded as inconvenient [4], so newer systems integrate directly with a device which the user would already be carrying, such as a smartphone. Strong reasons for the widespread adoption of tap-and-pay systems include usability and security [5]. Tap-and-pay systems require the user to install an app on a device and provision a payment card to a virtual wallet. In Apple Pay[2], a token is created for each card and stored in the device's secure element; payments are made between the client and a compatible reader over NFC, with

[1] www.emvco.com/about_emvco.aspx (last accessed: June 2017).
[2] www.apple.com/business/docs/iOS_Security_Guide.pdf (last accessed: June 2017).

the user authenticating to the app using TouchID (fingerprint) or a passcode to authorise the payment. Android Pay[3] and Samsung Pay[4] are similar, but all card data processing and tokenisation is handled on a cloud server (due to the greater range of supported devices and their differing levels of security), so the client must connect to the server regularly to acquire new tokens; this has a negative impact on usability in areas where WiFi availability is poor.

It is difficult to prevent eavesdropping over invisible channels (*e.g.*, [6, 7]) and limiting the range to reduce the risk is not reliable (*e.g.*, [8, 9]). Some systems communicate over a visual channel between the client and reader, giving the user more control over the broadcast, potentially making it difficult for an adversary to intercept without being noticed. In Yoyo Wallet[5], a virtual wallet is hosted on a cloud server; the user authenticates to the app using a PIN, then a QR token is passed between the client and a specialist reader to authorise a payment. Each QR token may be used up to three times before the client needs to reconnect to the server, meaning it is more connection-dependent than tap-and-pay systems. Two similar systems, WeChat[6] and AliPay[7], both currently very popular in China, support QR codes and barcodes to transfer information.

There is little compatibility between newer payment systems, and exclusionary business models often mandate the use of specialist or dedicated point-of-sale readers. Merchants struggle to accept them all, so brand loyalty and local trends may factor into consumer decisions, detracting focus from security. Furthermore, the physical presence of a specialist reader may give a false perception of trust to a user that the reader is legitimate (a rogue reader could easily be dressed to look genuine). Anti-phishing systems exist in other forms [10, 11] and some banking interfaces (*e.g.*, DoubleSafe[8]) use a personalised greeting message to authenticate to the user before requesting a PIN or password, but we are yet to see this feature in point-of-sale readers—instead, we see measures such as payment limits, which mitigate damage at the expense of usability. Purnomo *et al.* [12] present a system where mutual authentication is achieved via a trusted third party, however it requires a connection throughout.

None of the systems offer an inherent mechanism whereby unauthorised usage is easily detected by the user, aside from manually checking the account balance. Yoyo Wallet comes close: if an adversary were to expend the three uses of a stolen QR token, it would become useless and so indicate a problem when the user next tries to use it (unless the user is online, in which case the token will automatically refresh before use).

In this paper, due to space limitations, we focus on popular, real world systems. We identify five common drawbacks in these systems and propose a new mobile payment scheme with the goal of overcoming those drawbacks.

---

[3] support.google.com/androidpay (last accessed: June 2017).
[4] www.samsung.com/us/support/answer/ANS00043790 (last accessed: June 2017).
[5] www.yoyowallet.com/support.html (last accessed: June 2017).
[6] pay.weixin.qq.com/index.php/public/wechatpay (last accessed: June 2017).
[7] global.alipay.com/products/spot (last accessed: June 2017).
[8] www.tangerine.ca/en/security (last accessed: Oct. 2017).

2 Objectives and Assumptions

Design Objectives. The purpose of our system is to authenticate a user to a verifier using a client and via a point-of-sale reader to authorise a payment; the system should also provide the following features to overcome the common drawbacks identified in existing systems:

- *No specialist hardware requirement.*
- *Mutual authentication:* the system should authenticate the user to the verifier via the reader; it should also authenticate the verifier and the reader to the user *before* he authenticates to it, so as not to reveal secrets to a rogue reader.
- *Visual channel:* the system should operate over a visual channel between the client and the reader.
- *No client-to-verifier connection requirement.*
- *Unauthorised usage detection:* the user should be told if an unauthorised user has impersonated him, in a way that the intruder cannot avoid or erase.

System Model. The system consists of four components: a *user* (prover); a *client*—*i.e.*, a user device, such as a smartphone, with a camera, a screen, and our app installed on it; a *verifier*, such as an authentication server, which maintains a database of users' cryptographic materials (see §3); and a point-of-sale *reader* with a camera, a screen, and a means to enter a PIN (such as a touchscreen).

During enrollment, we assume that there is a secure channel between the client and the verifier over which they exchange cryptographic materials. The user will choose a PIN and a personalised message; we assume the former can be reset only by connecting with the verifier, whereas the latter can be changed for freshness on the client at any time without connecting to the verifier.

During authentication, we assume that there is a visual channel between the client and the reader, and a secure channel between the reader and the verifier. The client captures the payment amount and embeds data into an image to transfer it over the visual channel. For any given user, we assume that only one authentication attempt may be active at a time; simultaneous attempts should be rejected by the verifier. A visualisation of the system is shown in Figure 1.

The system verifies three factors to authenticate the user: possession of the client (something he has), verified by the data it embeds into the image, and knowledge of the image and PIN (something he knows). It is recommended, but not assumed, that the client require the user to authenticate to it using a biometric, such as a fingerprint, to add a fourth factor (something he is).

Threat Model. We assume that the adversary can watch and modify communications between the user and the verifier, such as by deploying a rogue reader. We assume that he knows everything about the user and may have access to any of the images used—*e.g.*, the user may have shared them on social media, with or without embedded data.

The goal of the adversary is to impersonate a legitimate user and either authorise or modify a payment without that user's knowledge. In the first case,

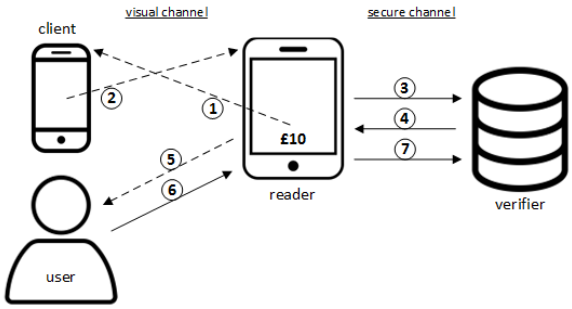

Fig. 1. The system model. The client reads the payment amount (1), embeds the amount, a message, and some authentication data into an image, and displays the image to the reader (2), which sends it to the verifier (3). The verifier returns the message (4), which authenticates it to the user (5), who then enters his PIN (6), which authenticates him to the verifier (7).

the adversary is a rogue user who may attempt to perform a *replay attack*, by re-using a previous image token to authorise a new payment. In the second case, the adversary is a rogue merchant who may attempt to perform a *tampering attack*, by using a rogue reader to authorise a payment for an amount different to what is displayed by modifying the image token.

In this paper, we will not consider attacks on the client, such as physical theft, cloning, or malware—these are all to be covered in future work. We also do not consider attacks that take place during the enrollment phase, attacks on the verifier, or denial-of-service attacks.

3 System Architecture

Cryptographic Materials. During enrollment, the verifier exchanges some cryptographic materials with the client. It shares its public key, I_e, and its public symmetric transposition key, M, used for embedding. It generates the user a unique identifier, ID, and two secret block cipher keys[9], K, L: K is used by the client to encrypt data while embedding it into the cover-image; L is not shared with the client. It also generates the user two blocks of secret binary data, p, u: p is used to authenticate the client to the verifier and its size should be sufficiently large to authenticate with confidence (*e.g.*, 1,000 bits); u is used to update K and p after each authentication and its size should be large enough to cover both.

The user chooses a secret PIN of memorable size (*e.g.*, 4 digits); PIN is stored on the verifier and $\{PIN\}_L$ is returned and stored on the client. The user also chooses a personalised greeting message, m, used to authenticate the reader and the verifier to the user; it need not be remembered, only recognised, but its size should be bounded to fit on a reader's screen (*e.g.*, up to 40 characters).

A summary of the materials used in the system is shown in Table 1.

[9] An authenticated encryption algorithm should be chosen, such as AES-EAX.

Table 1. A summary of the materials used in the system.

	Stored on Verifier	Stored on Client	Purpose
ID	✓	✓	identifies user
I_e	✓	✓	verifier's public key
I_d	✓	×	verifier's private key
M	✓	✓	verifier's public transposition key
K	✓	✓	secret key; used to encrypt a, m, p
L	✓	×	secret key; used to encrypt PIN
p	✓	✓	authenticates client to verifier
u	✓	✓	updates K, p
H	✓	✓	hash function; modifies u
PIN	✓	×	authenticates user to verifier
m	×	✓	authenticates verifier and reader to user
a	×	×	payment amount

Embedding Data. In this paper, we will restrict our attention to embedding data in the spatial domain with a simple LSB-embedding algorithm. To embed some data d into an image using a transposition cipher M, we apply M to the image to rearrange its pixels, then we embed d into them sequentially; we denote this by $[d]_M$. After embedding, we recreate the image by inverting the rearrangement. To extract d from the embedded image, we reapply the rearrangement.

To protect the confidentiality of the data as it passes over the visual channel, we encrypt it before embedding it using some key k; we denote this by $[\{d\}_k]_M$.

To protect the integrity of the embedded data as it passes over the visual channel, we strengthen the algorithm with repeat embedding. To do so, we choose an odd number $n > 1$ and then embed $\{d\}_k$ into the image n times sequentially; we denote this by $[\{d\}_k^n]_M$. When extracting, we treat any discrepant bit as having whichever value was extracted for it most frequently.

Image and Storage. To use the client, the user must authenticate to it by selecting the correct cover-image from a set containing decoy images (beneficially, this is more human-usable than recalling a password [13]). To prevent an adversary from identifying the cover-image by metadata examination, we embed junk data into the decoys whenever the cover-image is updated.

We will store ID and $\{PIN\}_L$ embedded in the cover-image, such that an adversary with access to the client would need to identify it to find them. For this embedding, we can either use M or define a local key. Other cryptographic materials are stored securely within the client, using a secure element if available.

Unauthorised Usage Detection. After authentication, we update K and p for freshness; this provides resistance to replay attacks by making each embedded image good for only one use. To do so, we modify u using a hash function, H, which (i) preserves the size of u and (ii) ensures that its future values are not predictable; we use a to achieve the latter, since a is known to both the client and the verifier at the time of hashing,

$$u = H(u \| a).$$

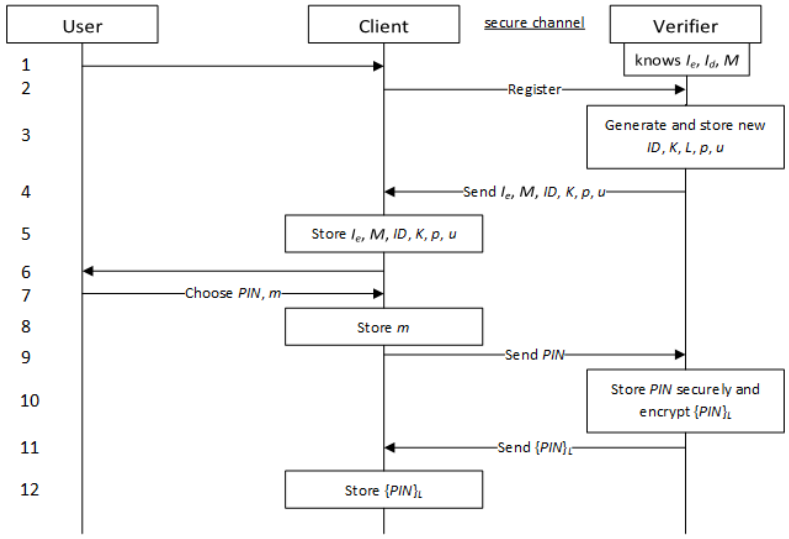

Fig. 2. The enrollment protocol.

We then apply our new u as a stream cipher to update K and p,

$$\{K, p\} = u \oplus \{K, p\}.$$

We do this on both the client and the verifier to keep the values synchronised.

By updating K and p after each authentication, we achieve inherent unauthorised usage detection. If an adversary were to successfully impersonate the user, the K and p values on his client and the verifier would update, unavoidably de-synchronising the user's client's K and p values from the verifier's. The system can be reset by exchanging new cryptographic materials with the verifier over a secure channel. In the case of unauthorised usage, the verifier can identify the last legitimate transaction by using the client's value of u and recreating the verifier's current value of u, since the latter is the result of deterministic hashes of the former with ordered payment values known to the verifier.

Enrollment Protocol. A secure channel is established between the client and the verifier to register an account and exchange cryptographic materials, such that the user can later authenticate. The protocol is shown in Figure 2.

Steps 1-5. When the user registers an account, the verifier generates him a new ID, K, L, p, and u and stores them. The verifier knows M, I_e, and I_d and has values set for n and H. The verifier sends I_e, M, ID, K, p, and u to the client, which stores them.

Steps 6-12. The client prompts the user to choose PIN and m; it stores m, sends PIN to the verifier, and gets $\{PIN\}_L$ back, which it stores.

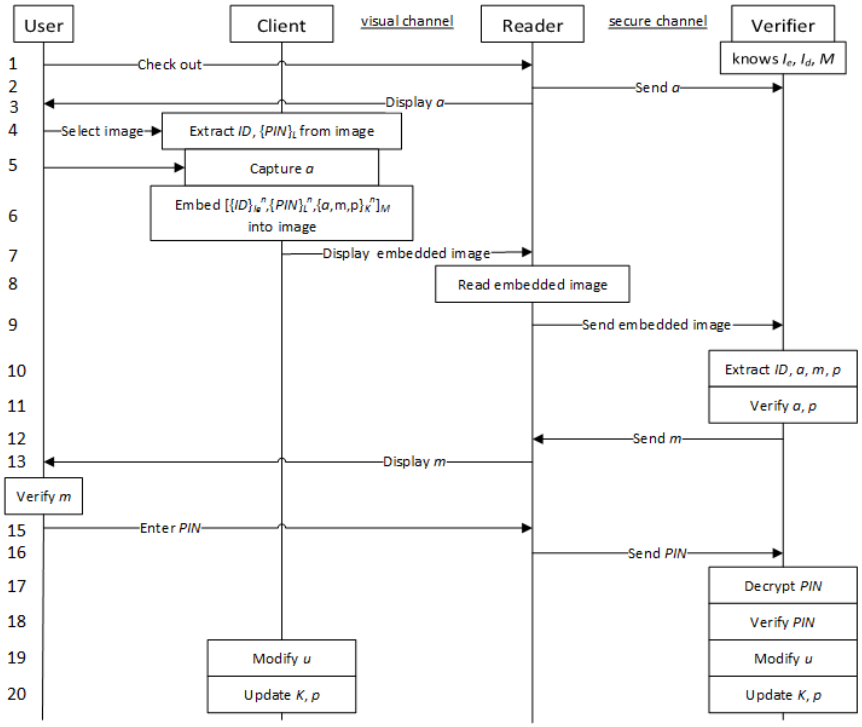

Fig. 3. The authentication protocol.

Authentication Protocol. A visual channel between the client and the reader and a secure channel between the reader and the verifier are required for the system to achieve mutual authentication. For any given user, only one authentication attempt may be active at a time. The protocol is shown in Figure 3.

Steps 1-6: User Authenticates to Client. The user authenticates to the client by selecting the cover-image; the client extracts ID and $\{PIN\}_L$. The reader sends a to the verifier to initiate the transaction and displays a. The client captures a using its camera and embeds $[\{ID\}_{I_e}^n, \{PIN\}_L^n, \{a,m,p\}_K^n]_M$ into the image to create the image token. The user may change m before embedding the data, since the required keys are stored on the client.

Steps 7-11: Client Authenticates to Verifier. In a commitment scheme [14], the user displays the image token to the reader, which sends it to the verifier. Firstly, the verifier knows M and so extracts $\{ID\}_{I_e}$, $\{PIN\}_L$, and $\{a,m,p\}_K$. Secondly, it uses ID to look up K to decrypt a, m, and p. Thirdly, it verifies that a and p match its expectations; this authenticates the client to the verifier.

Steps 12-14: Reader and Verifier Authenticate to User. The reader and verifier authenticate to the user (and confirm a) by displaying m on the reader.

Steps 15-18: User Authenticates to Verifier. The user enters PIN to the reader, which sends it to the verifier. The verifier compares the entered PIN,

the $\{PIN\}_L$ extracted from the image token, and its own stored version to ensure that all three of them match; this authenticates the user to the verifier.

Steps 19-20: Client and Verifier Modify u and Update K and p. The verifier modifies u using a hash function $H(u\|a)$ and uses u to update its K and p values. The client does likewise such that their respective values remain the same.

4 Discussion

The system provides real-time authentication and achieves its design objectives. Firstly, it requires no specialist hardware: the reader needs only a camera, a screen, and a touchscreen, which can be satisfied by any modern smartphone or tablet, making it easily deployable. Secondly, the authentication protocol provides resistance to phishing attacks in the form of mutual authentication by ensuring that the user authenticates to the client, the client to the verifier, the reader and verifier to the user (before PIN needs to be revealed), and the user to the verifier. The system authenticates the user to the verifier by transferring information, embedded in an image token, over a visual channel, meaning that it does not require a client-to-verifier connection nor the use of invisible channels, such as NFC, which risk interception. The user is informed of unauthorised usage as an inherent part of the protocol. Furthermore, since the image token contains the payment amount, the system does not only authenticate the user (which might be misused), but also binds the amount to be authorised.

The system relies on the user in two ways. Firstly, it is conceivable that a poor choice of cover-image weakens user-to-client authentication [15]; this reliance is alleviated if a biometric is used to authenticate the user to the client. Secondly, one benefit of using a visual channel over an invisible channel is that the user, with reasonable care, has greater control over who or what can see the image token; however, such care cannot be reasonably expected of all users.

For the *replay attack*, the adversary attempts to impersonate a legitimate user to authorise a payment by replaying a captured image token that was previously sent to the verifier. At the end of the transaction from which it was captured, the data embedded in the image token became out-of-date. In order to update it, the adversary would need to know K, p, u, and a. While the adversary may know a, and then be able to determine K and p by brute force, u is not stored in the image at all. To glean any useful knowledge of u by observing its effects over time would require an impractical number of transactions. Therefore, the system is resistant to replay attacks.

For the *tampering attack*, the adversary attempts to have a legitimate user authorise a payment for a different amount $a' \neq a$ by using a rogue reader to modify the embedded data in the image token before sending it to the verifier. At step 2 of the authentication protocol, the adversary sends a' to the verifier while displaying a to the user at step 3. At step 8, the adversary can extract $\{a, m, p\}_K$ from the image since M is public; however, he would need to compromise the secret key, K, to change a or the attack will fail at step 11 when the verifier decrypts and compares it with a' from step 2. Assuming K is a strong,

authenticated encryption key, then it is unlikely that the adversary will be able to compromise it in the short time available before the user becomes suspicious. Therefore, the system is resistant to tampering attacks.

5 Conclusion and Future Work

Our system meets its design objectives by providing a novel means to mutually authenticate a user, reader, and verifier over a visual channel without any specialist hardware nor a connection between client and verifier; it is resistant to replay and tampering attacks and offers inherent and unavoidable unauthorised usage detection. It makes use of three factors in a convenient manner: a device which would be carried anyway, a chosen image familiar to the user, and a PIN.

In future work, we intend to investigate attacks on the client, including physical theft, cloning, and malware. We note that PIN is not stored on the client, meaning that the system provides some resistance to indiscriminate malware infections; the adversary would need to observe the user entering PIN separately in each case, increasing the work required in such an attack. We also plan to run user studies to better understand user requirements, such as acceptable transaction durations and system intuitiveness, and to study in greater detail the technical constraints of using an embedded image, such as allowable noise and the use of gridlines to handle rotation.

References

1. British Retail Consortium. *Debit Cards Overtake Cash to Become Number One Payment Method in the UK*. 2017.
2. Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei Skorobogatov, and Ross Anderson. *Chip and Skim: Cloning EMV Cards with the Pre-play Attack*. IEEE Symposium on Security and Privacy (SP), 2014.
3. Martin Emms, Budi Arief, Leo Freitas, Joseph Hannon, and Aad van Moorsel. *Harvesting High Value Foreign Currency Transactions from EMV Contactless Credit Cards Without the PIN*. ACM Conference on Computer and Communications Security (CCS), 2014.
4. Jupiter Research. *Integrated Handsets: Balancing Device Functionality with Consumer Desires*. 2005.
5. Jun Ho Huh, Saurabh Verma, Swathi Sri V Rayala, Rakesh B Bobba, Konstantin Beznosov, and Hyoungshick Kim. *I Dont Use Apple Pay Because Its Less Secure...: Perception of Security and Usability in Mobile Tap-and-Pay*. Proceedings of the Workshop on Usable Security (USEC), 2017.
6. Steven J Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. *Chip and PIN is Broken*. IEEE Symposium on Security and Privacy (SP), 2010.
7. Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. *On the security issues of NFC enabled mobile phones*. International Journal of Internet Technology and Secured Transactions, 2010.
8. Henning Kortvedt and S Mjolsnes. *Eavesdropping near field communication*. The Norwegian Information Security Conference (NISK), 2009.

9. Thomas P. Diakos, Johann A. Briffa, Tim W. C. Brown, Stephan Wesemeyer. *Eavesdropping near-field contactless payments: a quantitative analysis*. The Journal of Engineering, 2013.
10. Stuart E. Schechter, Rachna Dhamija, Andy Ozment, and Ian Fischer. *The Emperor's New Security Indicators*. IEEE Symposium on Security and Privacy, 2007.
11. Claudio Marforio, Ramya J. Masti, Claudio Soriente, Kari Kostiainen, and Srdjan apkun. *Evaluation of Personalized Security Indicators as an Anti-Phishing Mechanism for Smartphone Applications*. CHI Conference on Human Factors in Computing Systems, pp. 540-551, 2016.
12. Ariana T. Purnomo, Yudi S. Gondokaryono, Chang-Soo Kim. *Mutual authentication in securing mobile payment system using encrypted QR code based on public key infrastructure*. IEEE 6th International Conference on System Engineering and Technology (ICSET), 2016.
13. Robert Biddle, Sonia Chiasson, and P. C. van Oorschot. *Graphical Passwords: Learning from the First Twelve Years*. ACM Computing Surveys (CSUR), vol. 44, 2012.
14. Gilles Brassard, David Chaum, and Claude Crepeau. *Minimum Disclosure Proofs of Knowledge*. Journal of Computer and System Sciences, vol. 37, pp. 156189, 1988.
15. Darren Davis, Fabian Monrose, and Michael K. Reiter. *On User Choice in Graphical Password Schemes*. USENIX Security Symposium, vol. 13, pp. 11-11, 2004.