Key exchange with the help of a public ledger

Thanh Bui, Tuomas Aura

Aalto University
thanh.bui@aalto.fi, tuomas.aura@aalto.fi

Abstract. Blockchains and other public ledger structures promise a new way to create globally consistent event logs and other records. We make use of this consistency property to detect and prevent man-in-the-middle attacks in a key exchange such as Diffie-Hellman or ECDH. Essentially, the MitM attack creates an inconsistency in the world views of the two honest parties, and they can detect it with the help of the ledger. Thus, there is no need for prior knowledge or trusted third parties apart from the distributed ledger. To prevent impersonation attacks, we require user interaction. It appears that, in some applications, the required user interaction is reduced in comparison to other user-assisted key-exchange protocols.

1 Introduction

Authentication is essential in any key exchange over insecure network. It helps each party be certain that the exchanged key is in fact shared with the desired second party, not an impostor. While authenticated key exchange has been widely studied, it is still a task with many complications. Most of the existing key-exchange protocols require some root of trust between the involved parties, such as prior knowledge on a cryptographic key or password [9,11,15,20,23,40], or trusted third-parties (TTP) [14,34]. In situations where these roots of trust are not available, such as device pairing, key exchanges usually rely on user interaction to achieve authentication [5,9,17,21,22,26,38]. The probability of a successful attack against these protocols mostly depends on the entropy of the user input. While more complex inputs bring higher security level, they might also cause usability issues.

To ease the requirements of a secure key exchange, we find the solution in blockchains and similar *public ledger* structures, which provide a new way to publish information and achieve consistency in a distributed system. They have been used as key directories that store bindings between identities and public keys [1,24,29,41]. In a public-key-based key exchange, a party can look up for the public key of the other in these directories. While these systems provide a reasonable user experience and do not depend on any single TTP, they restrict a party to use only the registered keys and thus, complicate key management. Furthermore, they require all interested parties to have a well-defined and unique identity, which does not always hold in reality. For example, two mobile devices of the same user may share the same identity, such as the user's email address,

or they may be identified simply by their presence in the user's hands. In any case, the key-identity binding requires the user or someone else to register the bindings.

In this paper, we present a new family of key exchange protocols that utilizes the global consistency property of the public ledgers. We refer to the protocols as the *public-ledger-based (PLB) protocols.* The idea is to bring transparency to a key exchange by publishing its parameters into the public ledgers. This enables the communication parties to check whether they are participating in the same key exchange, thereby detecting man-in-the-middle (MitM) attacks. Our protocols do not require any roots of trust between the parties or key-identifier bindings. In cases where the parties have public identifiers, such as phone numbers or emails, MitM attacks are prevented without any user interaction. When public identifiers are not available, our protocols still require the users to act as an out-of-band channel. However, the amount of information conveyed via the out-of-band channel is not a function of the desired security level. Thus, high security level can be achieved with little user interaction. Of course, instead of being MitM, the attacker could perform an impersonation attack against one of the parties. We will present application-specific solutions to the attack.

The rest of the paper is structured as follows. Section 2 covers the background about public ledgers. Section 3 gives an overview of our solution, and Section 4 describes the PLB protocols in detail. Section 5 presents a number of practical applications of the protocols. Section 6 discusses the denial-of-service attacks against the protocols and the privacy issues and presents some solutions. Section 7 concludes the paper.

2 Background

This section covers the literature about public ledgers and the threat model that we consider in this paper.

2.1 Public ledger

The recent widely known compromises of popular service providers [2,3,4] and government surveillances [16,18,28] have motivated quite a few proposals for public ledgers that have no centralized management. These public ledgers are basically public logs of *events* with the following properties [12]: *immutability* and *global consistency* of the event history, *inclusiveness* in the sense that all valid data is included. The ledgers might also define a *linear order* or have some built-in concept of time on all data entries.

Two main architectures have emerged for the public ledgers [13]: (1) totally distributed with no central points of trust or (2) more centralized with one untrusted third party who manages the ledger's content and several trusted auditors who audits its behaviors.

Distributed ledgers. The most prominent representative of the distributed ledgers is Bitcoin [30]. In Bitcoin, an open peer-to-peer (P2P) network with no

single party responsible for any critical operation maintains an append-only log of transactions, called *blockchain*. Transactions take place between public keys and are communicated as signed messages in the P2P network. They are mined into blocks approximately every ten minutes. Global consistency is achieved by compressing the global history of transactions into one cryptographic hash value, which can relatively easily be agreed on and communicated to everyone. Inclusiveness and immutability are guaranteed by a novel distributed consensus mechanism and a competitive mining process.

The Bitcoin's blockchain has become a reliable log for many other applications, such as CommitCoin [4], Factom[1], Proof of Existence[2], Virtual notary[3], and Tierion[4]. It has also inspired hundreds of alternative cryptocurrencies, such as Ethereum [39] and Namecoin [24]. A number of central banks and financial institutions have started investing in Bitcoin-based technologies [7,35,31].

Centralized ledgers. Although Bitcoin and its variants have achieved a large degree of success, the decentralized nature limits their ability to achieve widespread adoption. The reason is that, for the systems to be completely trustless, users must store the entire blockchain locally and track its progress. Thus, the amount of storage and communication traffic that they require increases linearly with the number of users.

To address the shortcomings of the blockchain, more centralized ledger architectures have been proposed in the literature, in which the ledger content is maintained by an untrusted third party instead of the P2P network. They rely on trusted *auditors* to achieve the desired properties.

Most of the proposals in this category are for monitoring security of the web PKI. Certificate Transparency (CT) [27] is a public log of all web certificates, which aims to bring transparency and accountability to the CA operations. The log is structured as an *append-only Merkle tree*, in which new records are added to the right of the tree. Accountable Key Infrastructure (AKI) [25] and its kin [8,37] are similar solutions, but they deploy an *ordered Merkle tree*, where the data in the leaf nodes is sorted by the domain name instead. Both append-only and ordered Merkle trees enable logarithmic-size proofs of existence and non-existence for certificates, though. In yet another variant of these ideas, PKI Safety Net [36] enables verification of both the issuing time order and the non-existence of records for a domain by maintaining two trees that are similar to those in CT and AKI, respectively. Independent *monitors* maintain a copy of the trees and audit their consistency.

Apart from web PKI, centralized ledger solutions have been suggested for other applications. CONIKS [29] constructs directories of user certificates. DECIM [41] keeps track of uses of a public key in a transparency log so that its owner can detect key misuse. Enhanced Certificate Transparency [33] extends CT to handle certificate revocation and shows how this extension can be used in end-to-end

---

[1] https://www.factom.com/
[2] https://proofofexistence.com/
[3] https://virtual-notary.org/
[4] https://tierion.com/

email or messaging systems. Bui et al. [12] apply public ledgers to group management in distributed systems where entities are represented by their public keys and authorization is in the form of signed certificates.

In this paper, we are not proposing any new ledger structure. Instead, *we assume that a public ledger with the properties mentioned above exists and we use it as an abstract to build our key exchange protocols.*

2.2 Threat model

We consider a threat model where there are active attackers against key exchanges. The attackers can perform *man-in-the-middle (MitM)* attacks to intercept the communication traffic or *impersonation* attacks to pretend to be one of the involved parties. They can also perform *denial-of-service (DoS)* attacks to cause failures to key exchanges.

We assume that all the underlying cryptography is secure. Also, the communication channel between any party and the ledger, if exists, is secured under existing security protocols such as TLS/SSL.

3 Basic idea

The key observation that motivated this work is that *in a key exchange that is free of MitM, the endpoints have a consistent view of the key exchange's parameters (i.e. the public keys of the endpoints or the shared key).* Suppose that when Alice wants to establish a shared key with Bob, Carol performs a MitM attack to intercept the communication. If Carol succeeds, there will be two different key exchanges — one between Alice and Carol and the other between Carol and Bob, as illustrated in Figure 1. A secure connection, on the other hand, involves only a single key exchange. In other words, the MitM attack creates inconsistent views of the key exchange between Alice and Bob, while in the case of a secure key exchange, their views are consistent.
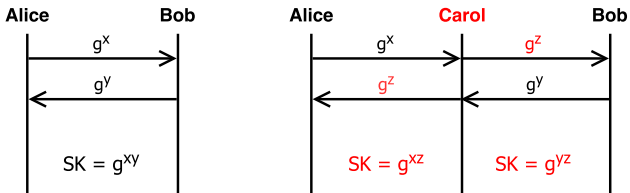

Fig. 1. A secure key exchange (on the left) vs a key exchange with MitM attacker (on the right)

Based on the above observation, *our proposal for detecting MitM attacks against any unauthenticated key exchange is to publish its parameters to a public*

ledger. Let us assume such a ledger exists and (unrealistically) that there is only one legitimate key exchange taking place in the world ever. After the unauthenticated key exchange, one of the parties (e.g., the initiator) sends an event E containing the key exchange's data to the ledger, which can be, for example, a hash of the public keys or a hash of the shared key. When the ledger has made the event public, the parties separately query the ledger to check how many new events were entered into the ledger. If there is only one and it matches their view of the key exchange, the key exchange was secure. On the other hand, if two or more events were entered, there may be a MitM attacker around and both parties must discard the results of the key exchange.

Context. The scheme described above is impractical because it supports only one key exchange ever. To enable multiple key exchanges, our first attempt is to amend the key-exchange event E with a context C. The data submitted to the public ledger will be $\langle C, E \rangle$ and it is indexed by C. That is, one side submits this data to the ledger, both sides query it with the index C, and then they compare the received event E (if there is only one) with their view of the key exchange.

The context can be any information that the two parties agree on *naturally* or *out of band*. The following information is particularly suitable to be used in C:

- application identifier a,
- endpoint identifier i if one is already known to the other party,
- user input or user-compared code c.

The first and second elements are examples of natural contexts, while the last is an example of out-of-band contexts. A key exchange can use both natural context and out-of-band context. How much information the context should hold depends on how many parties are simultaneously writing to the ledger. The frequency of collisions for the index C should be sufficiently small to not frustrate the users.

Time window. We can see that by tagging each key exchange event with a context, once a context is used, it cannot be re-used for any other key exchange. To optimize the solution, we tag an event E not only with a context C but also with a time t. This way, we can do one key exchange per context per *time period* τ. Defining the time period is application-specific, and it must be agreed by the two parties in advance. An application, for example, can define that a context can be used once per minute or per hour.

We can consider time t as a part of the context C. However, it must be provided by the ledger because we cannot trust the communication parties to timestamp the key exchange events. Thus, the ledger must have a built-in concept of time. In Bitcoin, for example, a new block appears every ten minutes; thus, we can roughly determine the timestamp of a block by checking how many blocks are after it in the block chain.

Figure 2 illustrates an overview of the protocol with the addition of the context and the time period. The data submitted to the ledger will be $\langle C, E \rangle$
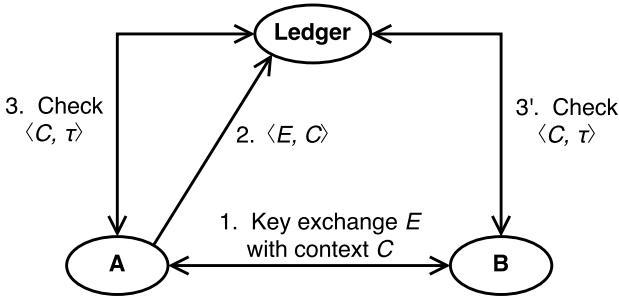
Fig. 2. An overview of the public-ledger-based key exchange protocol

and it is timestamped by the ledger. When each party checks the ledger, they only query for events with context C that happened in the time period τ. The parties reject the key exchange if there is more than one events that matches this query or if the retrieved event E does not match theirs.

4 Public-ledger-based key exchange

We now describe our PLB key exchange protocols in detail. We will start with a protocol where the communication parties agree on a context out of band and user interaction is required for the agreement. We will then build on it a protocol where the two communication parties must naturally agree on a context with each other and no user interaction is involved.

4.1 Public-ledger-based key exchange with out-of-band context

Since the PLB key exchange protocol with out-of-band context requires user interaction, we require the communication parties to be equipped with the following: (1) a screen that can display a string, and (2) a "start" button where users can press to trigger an event.

The protocol involves four parties: the initiator A, the responder B, the ledger L, and the user U. It consists of six phases as follows.

1. **Unauthenticated key exchange**: In the first phase, the parties perform an unauthenticated key exchange to establish a shared secret key SK.
2. **Context acquirement**: The initator A requests the ledger for a context that is not in use. The ledger then returns a context C that matches the query. A then sends the context to B using the shared key. (An alternative way to acquire a context is for one party to generate it locally and try to use it. This way, depending on the entropy of the context, collision might happen, causing failure in the protocol.)
3. **Context comparison**: A and B display the context C on their screen so that user U can compare the context. If they display the same context, U

presses "start" on both endpoints at the same time. After that, both parties immediately query the ledger to get the current ledger's time. Let S_A and S_B be the ledger's times that A and B receive, respectively.

4. **Commitment**: A commits the key exchange to the ledger. It submits an event E containing the key exchange's parameters along with the context C to the ledger. The ledger then stores E and indexes it by C. It also gives the event a timestamp t. If for some reason (e.g., network failures) the submission fails, A aborts the protocol and informs B about the abortion. Otherwise, A informs B that the commitment was successful via the encrypted channel established in the previous phase.

5. **Commitment verification**: When the ledger has published the event in the previous phase, both parties separately query the ledger for events with context C. A queries for events that happened only in period $[S_A - \triangle, Q_A]$, and B queries for events that happened only in period $[S_B - \triangle, Q_B]$, where Q_A and Q_B are the times when A and B make the query, respectively, and $\triangle$ is an application-specific constant that represents the possible difference in clocks between A and B. The ledger then returns a list of events that match the query and a proof that the list is complete. For each event in the list, the ledger also returns the respective timestamp. The parties verify the list. They abort if any of the following occurs: (1) the verification fails, (2) more than one event appear in the list, or (3) the only event in the list does not match their view. Otherwise, the protocol succeeds and the parties accept the shared key.

6. **Confirmation**: User U checks that the protocol succeeds on both sides. It it fails on either side, the user must abort the protocol.
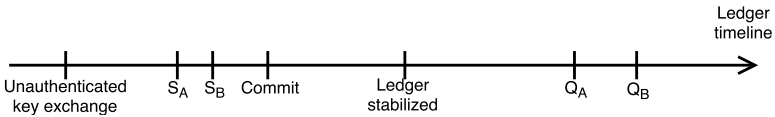

Fig. 3. Ledger's timeline of events for a PLB protocol using out-of-band context

Figure 3 illustrates all the events in the key exchange above in the ledger's timeline. We can see that if there is man in the middle, the party who queries last (B in the figure) will see that there are two key exchanges with the same context C and thus, abort the protocol. Impersonation attack is also not possible because the user will check that the protocol succeeds on both sides.

A notable property of the protocol is that its security level does not depend on the entropy of the context. In order for the protocol to succeed, each key exchange simply needs a unique context during its time period. How much information the context should hold depends only on how many parties are simultaneously writing to the ledger.

4.2 Public-ledger-based key exchange with natural context

In many cases, the user cannot be present at both sides to press the "start" button as in the previous protocol. However, the communication parties in these cases usually have a public identifier that is already known to each other. For example, in an encrypted phone call, the identifiers are the phone numbers or the email addresses. These identifiers can form a context on which the parties can naturally agree with each other. This section will present the PLB key exchange when a natural context is available. This protocol is more simple but requires more assumptions than the previous does because there is no user interaction.

The PLB key exchange with natural context involves three entities: the initiator A, the responder B, and the ledger L. Beside a context C, the initiator and the responder need to agree with each other on the following parameters in advance:

- A time period's length w, which indicates that a context can only be used once in a time period of length w. The first time period of a day starts at 00:00. We assume that the local clocks of the communication parties must be synchronized with reliable sources, and w must be larger than the difference between these clocks.
- A timeout α, which indicates the maximum duration of a key exchange.

These parameters can be pre-configured in the communication application. This way, by using the same application to communicate with each other, the two parties naturally agree on the parameters.

The protocol consists of three phases: *unauthenticated key exchange*, *commitment*, and *commitment verification*. The details of each phase are as follows.

1. **Unauthenticated key exchange**: As in the previous protocol, the parties first perform an unauthenticated key exchange to establish a shared secret key SK. Suppose that the current time period is $[t_0, t_0 + w]$ and the current time is t. The parties can only perform the unauthenticated key exchange if t is not within $[t_0, t_0 + \alpha]$ or $[t_0 + w - \alpha, t_0 + w]$. This is to ensure that the key exchange does not span over two time periods. Let S_A and S_B be the times when the key exchange starts on A and B, respectively. A will abort the protocol if it does not finish by time $S_A + \alpha$, and B will do so by time $S_B + \alpha$.

2. **Commitment**: The initiator A submits an event E containing the key exchange's parameters along with the context C to the ledger. The ledger then stores E and indexes it by C. It also gives the event a timestamp t. If for some reason (e.g., network failures) the submission fails, A aborts the protocol and informs B about the abortion. Otherwise, A informs B that the commitment was successful via the encrypted channel established in the previous phase.

3. **Commitment verification**: When the ledger has published the event in the previous phase, both parties separately query the ledger for all events with context C in the current period. The ledger then returns a list of events

that match the query and a proof that the list is complete. For each event in the list, the ledger also returns the respective timestamp. The parties verify the list. They abort if any of the following occurs: (1) the verification fails, (2) more than one event appear in the list, or (3) the only event in the list does not match their view. Otherwise, the protocol succeeds and the parties accept the shared key.

The protocol described above can prevent MitM attacks as follows. Assuming that there is a MitM attacker, the attacker must perform a separate key exchange with each party. However, the two key exchanges must share the same context C. With time synchronization and the timeout, we ensure that the communication parties perform the key exchange in the same time period. Thus, at least one of the parties will see two key exchanges with the same context published in the ledger and abort the protocol.

We can see that instead of being the MitM, the attacker could perform an impersonation attack against one of the honest parties (and denial-of-service against the other if necessary). Then, one party will be see the expected one key exchange in the ledger while the other will not even check. We will present application-specific solution to detect impersonation in the next section.

5 Applications

We now consider a number of practical applications of the PLB key exchange protocol.

5.1 Encrypted VoIP

We first present the most convincing application of the protocol — *encrypted voice over IP (VoIP)*, such as Cryptophone[5], SilentPhone[6], and Signal[7].

In these applications, each user is typically represented by a public-key pair, and the users use these key pairs to establish end-to-end encrypted phone calls. To verify that they are actually communicating with the parties they intend, the users can rely on trusted authorities who vouch for the users' public keys, such as the certificate authority in the X.509 Public Key Infrastructure. A major issue with trusted authorities is that they can vouch for fraudulent keys in an attack [3,2]. An alternative approach is that the users perform key verification out of band. A common approach for key verification is to compare key fingerprint, i.e. some representation of a cryptographic hash of the public keys. Key fingerprint comparison introduces severe usability because users have to manually perform key verification before communicating with a new partner. Shorter strings can be provided instead to simplify the verification [10,42]. A short authentication

---

[5] http://www.cryptophone.de/
[6] https://www.silentcircle.com/
[7] https://whispersystems.org/

string (SAS) is a truncated cryptographic hash of the key exchange's parameters, which is usually represented by a human-friendly format. Participants compute the SAS based on the key exchange they observe and compare the resulting value by reading it aloud. The authentication is based on recognition of the other party's voice.

The PLB key exchange protocol can be applied readily to encrypted VoIP and improve the usability. Specifically, the parties perform the protocol with the context being the phone numbers (or user IDs). This context is inherent to both parties; thus, no out-of-band channel is needed for context verification. The use of the public ledger prevents MitM attacks, and the users will naturally start speaking secrets only if they recognize each other's voices. Therefore, the protocol achieves the same level of security as the SAS verification approach's but better usability because no user interaction is required.

Of course, users who have never heard each other's voice cannot rely on our protocol to establish a secure phone channel (same as with the SAS verification solution). Voice spoofing is also an issue of voice-based authentication [19,32]. Thus, we do not aim to completely replace the stronger methods of authentication, such as manual key fingerprint verification, but to complement them.

There have been attempts to publish bindings between user identities and public keys to public ledgers [1,24,29,41]. The main difference between these solutions to ours is that they restrict a party to use only the registered keys and thus, complicate key management. Our protocol, on the other hand, allows users to use arbitrary key pair for each communication, making the use of multiple devices with separate keys easier.

5.2 Device pairing

Device pairing — the procedure of establishing a shared key for secure communication between two devices — is another application of the PLB protocol. In device pairing, there is usually no key management infrastructure or pre-shared secret between the devices. Thus, authentication solely depends on some form of user action, such as entering a passkey [9,17,21] or verifying numeric code [5,22,26,38]. The security level of these approaches depend on the entropy of the user input. In Bluetooth, for example, users have to compare six digits displayed on the devices and the probability of a successful attack is 10^{-6}. Using longer or more complex inputs would be more error-prone and reduce usability.

The PLB key exchange protocol might offer advantages other the existing solutions in this scenario. The devices can run the PLB key exchange protocol with out-of-band context, and the users compare the context displayed on the devices. If they are equal, the key exchange succeeds with impossibility of MitM attacks, regardless of the entropy of the context.

The protocol can be applied to group association where more than two devices want to establish secure communication with each other.

6 Discussion

In this section, we discuss denial-of-service attacks against the PLB protocols as well as their privacy issues.

6.1 Denial-of-service

We have showed how the PLB key exchange protocols can defend against MitM and impersonation attacks. Another security vector to the protocols is the denial-of-service attacks where the attacker spams events to the ledger. This could prevent key exchanges to succeed because the endpoints could observe several key exchanges with the same context as theirs in the ledger and drop the key exchange as a result. In this section, we will analyze the causes of these attacks and how we can mitigate them.

The PLB key exchanges using natural contexts are attractive targets of DoS attacks. It is due to the fact that natural contexts are usually endpoint identifiers, which are known by everyone. To permanently prevent any pair of parties to communicate, an attacker can determine the context based on their identifiers and continuously send fake events with the context to the ledger. A solution for this type of targeted attacks is for the ledger to authenticate endpoint identifiers. Authentication could be easily adopted by centralized ledgers like Certificate Transparency. Also, the service provider can act as the ledger provider itself; thus, users can use the same credentials for the application and ledger.

The out-of-band contexts might also be guessed if their entropy is low. Thus, they must be random so that it is difficult to guess by the attackers but not too complex for users to compare. Using the ledger-provided contexts as in our protocol can limit the attack to some extent but cannot prevent massive spamming. Limitation on rate of submissions, posting fee (e.g. the transaction fee in Bitcoin), and proof-of-work of clients (e.g. client puzzels [6]), are some solutions to prevent massive spamming.

The ledger might be the source of context leak. Depending on the ledger's architecture, the attackers can read the contexts immediately when new events are entered the ledger and then post fake events with the same contexts. The blockchain with P2P network is an example of such designs, in which transactions are propagated through the network for a while before they are mined into blocks. Thus, to prevent context leak, the ledger could be designed so that it can only be queried for individual contexts and and the response for any individual query leaks no information about other events in the ledger. CONIKS's [29] is an example of such design. The ledger could also be round-based, meaning that events can be accepted in a round and only published in the next round.

6.2 Privacy

We now argue about the privacy issue of the PLB protocol that uses the communication parties' identifiers as the context. It is true that if a record is published in the ledger every time two parties communicate with each other, the protocol

has serious privacy issue. However, that is not how applications work in practice. Two communication parties need to perform a key exchange only once and save the result locally for later uses. They do not have to perform another key exchange unless one of them changes its public key. Authentication also helps to prevent privacy issues. For example, a party can only query the ledger for the records that are associated with contexts containing its identifier. Furthermore, unlike ledgers that store key bindings, the content of our ledger can be erased periodically without affecting any key exchanges. This not only reduces the footprint of the ledger but also might be of little help in preserving user privacy.

7   Conclusion

We have shown that it is possible to have authenticated key exchange protocols that take advantages of the global consistency property of blockchains and other public ledgers to detect and prevent man-in-the-middle attacks. While our protocol requires the users to act as an out-of-band channel, the amount of information conveyed out of band is not a function of the desired security level. Instead, it depends on the number of simultaneous key exchanges and thus, on the load of the public ledger; the out-of-band information simply ensures each key exchange gets its unique context string. Further work is required on the detailed requirements and design of the public ledger and on analysis of denial-of-service threats.

References

1. Blockstack. `https://blockstack.org`, [Dec. 2016]
2. Comodo fraud incident. `https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`, [Jan. 2017]
3. DigiNotar SSL certificate hack amounts to cyberwar, says expert. `https://www.theguardian.com/technology/2011/sep/05/diginotar-certificate-hack-cyberwar`, [Jan. 2017]
4. Yahoo hack: 1bn accounts compromised by biggest data breach in history. `https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached`, [Jan. 2017]
5. Alliance, Z., et al.: Zigbee specification (2006)
6. Aura, T., Nikander, P., Leiwo, J.: DOS-resistant authentication with client puzzles. In: International workshop on security protocols. pp. 170–177. Springer (2000)
7. Barrdear, J., Kumhof, M.: The macroeconomics of central bank issued digital currencies (2016)
8. Basin, D., Cremers, C., Kim, T.H.J., Perrig, A., Sasse, R., Szalachowski, P.: ARPKI: Attack resilient public-key infrastructure. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 382–393. ACM (2014)
9. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: Proceedings of the Computer Society Symposium on Research in Security and Privacy. pp. 72–84. IEEE (1992)

10. Blossom, E.: The VP1 protocol for voice privacy devices version 1.2 (1999)
11. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 156–171. Springer (2000)
12. Bui, T., Aura, T.: Application of public ledgers to revocation in distributed access control. `https://arxiv.org/abs/1608.06592`
13. Chase, M., Meiklejohn, S.: Transparency overlays and applications. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 168–179. ACM (2016)
14. Dierks, T.: The transport layer security (TLS) protocol version 1.2. RFC 5246 (2008)
15. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Designs, Codes and cryptography 2(2), 107–125 (1992)
16. Electronic Frontier Foundation: National security letters. `https://www.eff.org/issues/national-security-letters`, [Jul. 2015]
17. Gehrmann, C., Mitchell, C.J., Nyberg, K.: Manual authentication for wireless devices. RSA Cryptobytes 7(1), 29–37 (2004)
18. Gellman, B.: The FBI's secret scrutiny. `http://www.washingtonpost.com/wp-dyn/content/article/2005/11/05/AR2005110501366.html`, [Jul. 2015]
19. Gupta, P., Shmatikov, V.: Security analysis of voice-over-IP protocols. In: Proceedings of the Computer Security Foundations Symposium. pp. 49–63. IEEE (2007)
20. Hao, F., Ryan, P.: J-PAKE: Authenticated key exchange without PKI. In: Transactions on computational science XI, pp. 192–206. Springer (2010)
21. IEEE Standards 802.11 WG, Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications
22. IEEE Standards 802.15 1-2005, Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs) (2005)
23. Jablon, D.P.: Strong password-only authenticated key exchange. ACM SIGCOMM Computer Communication Review 26(5), 5–26 (1996)
24. Kalodner, H., Carlsten, M., Ellenbogen, P., Bonneau, J., Narayanan, A.: An empirical study of Namecoin and lessons for decentralized namespace design. In: Proceedings of the Workshop on the Economics of Information Security (WEIS) (2015)
25. Kim, T.H.J., Huang, L.S., Perring, A., Jackson, C., Gligor, V.: Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure. In: Proceedings of the 22nd international conference on World Wide Web. pp. 679–690 (2013)
26. Laur, S., Nyberg, K.: Efficient mutual data authentication using manually authenticated strings. In: International Conference on Cryptology and Network Security. pp. 90–107. Springer (2006)
27. Laurie, B., Langley, A., Kasper, E.: Certificate transparency. RFC 6962, IETF (2013)
28. Lichtblau, E.: Judge tells Apple to help unlock iPhone used by San Bernardino Gunman. `http://www.nytimes.com/2016/02/17/us/judge-tells-apple-to-help-unlock-san-bernardino-gunmans-iphone.html`, [Jul., 2015]
29. Melara, M.S., Blankstein, A., Bonneau, J., Felten, E.W., Freedman, M.J.: CONIKS: Bringing key transparency to end users. In: Proceedings of the USENIX Security Symposium. pp. 383–398 (2015)
30. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)

13

31. O'leary, D., D'agostino, V., Re, S.R., Burney, J., Hoffman, A.: Method and system for processing internet payments using the electronic funds transfer network (2013), uS Patent App. 13/789,826
32. Petraschek, M., Hoeher, T., Jung, O., Hlavacs, H., Gansterer, W.N.: Security and usability aspects of man-in-the-middle attacks on ZRTP. Journal of Universal Computer Science 14(5), 673–692 (2008)
33. Ryan, M.D.: Enhanced certificate transparency and end-to-end encrypted mail. In: Proceedings of the The Network and Distributed System Security Symposium (2014)
34. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Workshop on the Theory and Application of Cryptographic Techniques. pp. 47–53. Springer (1984)
35. Shin, L.: Canada has been experimenting with a digital fiat currency called CAD-COIN (2016), [Mar. 2017]
36. Szalachowski, P., Chuat, L., Perrig, A.: PKI Safety Net (PKISN): Addressing the too-big-to-be-revoked problem of the TLS ecosystem. In: 1st IEEE European Symposium on Security and Privacy (2016)
37. Szalachowski, P., Matsumoto, S., Perrig, A.: PoliCert: Secure and flexible TLS certificate management. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 406–417. ACM (2014)
38. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Annual International Cryptology Conference. pp. 309–326. Springer (2005)
39. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper (2014)
40. Wu, T.D.: The secure remote password protocol. In: Proceedings of the Internet Society Symposium on Network and Distributed System Security. vol. 98, pp. 97–111 (1998)
41. Yu, J., Ryan, M., Cremers, C.: DECIM: Detecting endpoint compromise in messaging. Tech. rep. (2015)
42. Zimmermann, P., Johnston, A., Callas, J.: ZRTP: Media path key agreement for unicast secure RTP. RFC 6189 (2011)