Ticket Transparency: Accountable Single Sign-On with Privacy-Preserving Public Logs*

Dawei Chu[1,2], Jingqiang Lin[1,2,3], Fengjun Li[4],
Xiaokun Zhang[5], Qiongxiao Wang[1,2,3], and Guangqi Liu[1,2,3]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, CHINA
[2] School of Cyber Security, University of Chinese Academy of Sciences, CHINA
[3] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, CHINA
[4] Department of Electrical Engineering and Computer Science, the University of Kansas, USA
[5] Academy of Opto-Electronics, Chinese Academy of Sciences, CHINA

Abstract. Single sign-on (SSO) is becoming more and more popular in the Internet. An SSO ticket issued by the identity provider (IdP) allows an entity to sign onto a relying party (RP) on behalf of the account enclosed in the ticket. To ensure its authenticity, an SSO ticket is digitally signed by the IdP and verified by the RP. However, recent security incidents indicate that a signing system (e.g., certification authority) might be compromised to sign fraudulent messages, even when it is well protected in accredited commercial systems. Comparing with certification authorities, the online signing components of IdPs are even more exposed to adversaries and thus more vulnerable to such threats in practice. This paper proposes *ticket transparency* to provide accountable SSO services with privacy-preserving public logs against potentially fraudulent tickets issued by a compromised IdP. With this scheme, an IdP-signed ticket is accepted by the RP only if it is recorded in the public logs. It enables a user to check all his tickets in the public logs and detect any fraudulent ticket issued without his participation or authorization. We integrate blind signatures, identity-based encryption and Bloom filters in the design, to balance transparency, privacy and efficiency in these security-enhanced SSO services. To the best of our knowledge, this is the first attempt to solve the security problems caused by potentially intruded or compromised IdPs in the SSO services.

Keywords: Accountability, Privacy, Single sign-on, Transparency, Trust management.

* This work was partially supported by National Natural Science Foundation of China (Award 61772518) and National Key RD Plan of China (Award 2017YFB0802100).
Jingqiang Lin (`linjingqiang@iie.ac.cn`) is the corresponding author.

1 Introduction

Recently, the single sign-on (SSO) service has become a popular identity management and authentication infrastructure in the Internet. It is a mechanism that permits an authorized user to sign onto many related but independent networked systems or applications (referred to as relying parties, or RPs) with a single action of authentication. It reduces the costs and risks not only for administering user accounts centrally from administrators' perspective, but also for personal account management from individual users' perspective. As a result, there is a proliferation of SSO schemes, such as OpenID Connect [50] and SAML with WS-Security [44] in SOAP [23], which allow users to leverage their existing accounts in popular identity providers (IdPs) such as Google, Microsoft, Facebook, PayPal, etc. A recent study shows that 6.3% of Alexa top 1 million websites support SSO in 2018 [18].

As a new promising identity management paradigm, SSO has been extensively studied in recent years. Several security vulnerabilities caused by various design and implementation flaws have been reported [34, 54, 57, 63, 64, 67]. While many security schemes and patches have been proposed to address SSO security problems, most of them, if not all, typically model IdPs as trusted third parties and thus focus on securing other components and their interactions in the SSO paradigm [3, 25, 66, 69]. However, it is also widely recognized that SSO suffers from the *single-point of failure* problem [18, 37, 58], especially when an IdP or an IdP account is compromised. In such circumstances, the trust to IdPs or the trust established between RPs and IdPs by the means of SSO *tickets* (e.g., ID tokens in OpenID or assertions in SAML) no longer last.

While compromised accounts, due to phishing and weak passwords, remain a prevalent security issue in various online services, several recent security incidents indicate that third-party providers should not be fully trusted by default or as expected. For example, professional CAs were reported being intruded or deceived to sign a large number of fraudulent TLS server certificates [12, 19, 42, 59], which contain fake yet well-formatted information about certificate holders. The signing components of the CAs are assumed well-guarded with defense-in-depth protection technologies, for example, they are typically built on hardware security modules (HSMs) and/or operate in isolated network segments. Comparing to them, we believe the online signing components of IdPs are exposed to a larger attack vector and therefore face more vulnerabilities and attacks in practice.

If an IdP is malicious or compromised, it can issue fraudulent SSO tickets for a victim user account, which allow the adversary to sign on to all related applications on behalf of that user without triggering any error or alarm. Moreover, [37] even proposed a novel attack that tricks RPs into using a malicious IdP for legitimate user accounts. Similarly, if a user's IdP account is compromised, the adversary will obtain legitimate SSO tickets issued by the trusted IdP and gain access to the victim's accounts on related RPs. To make things worse, once the trust between RPs and IdPs is established, the "local notion of identity is not strongly tied to the IdP's account access or control" [18]. This means even an ephemeral SSO ticket issued for a previously compromised IdP account could

be used to create a long-term token (e.g., a persistent cookie set by the RP) that allows the adversary to access the victim's account at the RP. Finally, many SSO schemes lack the functionality for the IdPs to universally revoke access to RPs' accounts created or accessed by a compromised IdP account in practice [18].

Therefore, it is critical to ensure accountability in SSO services that allows any individual user to track the SSO tickets issued under her IdP account, even if her IdP account was previously compromised or the IdP itself is not completely trusted. Meanwhile, this additional security mechanism for accountability should incur only a minimum modification to existing SSO service implementation and a minimum (computation and communication) cost at the IdPs, which typically serve a large number of users. The underlying trust problem in such a distributed service setting involving potentially unreliable third-party security service providers is similar to the security problem introduced by malicious or compromised CAs in the PKI ecosystem. Motivated by the popular certificate transparency (CT) [32] scheme, which is widely adopted to enhance the security of TLS certificates [1,24,45], we propose *ticket transparency* to address the problem caused by a raudulent SSO ticket, no matter it is issued by a compromised IdP or by a benign IdP for a compromised user account. Such fraudulent tickets are well-formatted and verifiable SSO tickets issued for an IdP account without the consent from the actual account holder.

Similar to certificate transparency, ticket transparency introduces log servers to monitor the ticket-signing operations of the IdPs. In particular, an IdP-signed ticket needs to be signed by a log server again, before it is accepted by the target RP. This additional signature by the log server is also a promise to record the SSO tickets in publicly-visible logs. Therefore, ticket transparency enables any user, who suspects an attacker signing onto any RP on behalf of her with a fraudulent ticket, to search for all tickets with his account in the public logs and detect the fraudulent ones among them. However, the certificate transparency framework cannot be directly adopted in SSO services due to the privacy concerns. Different from certificates, SSO tickets contain privacy information about the service requester (i.e., the user), the service provider (i.e., the RP), the occurrence time of sign-on activities, etc.

In order to protect user privacy while enable efficient ticket search in the accountable SSO services, we integrate blind signatures [11], identity-based encryption (IBE) [8], and Bloom filters [7] in the design of ticket transparency. First, a ticket is *blindly signed by the log server*, so the content of the ticket is protected against the log server. That is, the ticket is blinded before it is recorded in the public logs. Secondly, the secret blinding factor is stored along with the ticket in public logs, but *encrypted using the user's IBE public key* by the IdP. Therefore, only the user is able to un-blind his tickets. This encryption is identity-based, which reduces the overhead of key management in IdPs. More importantly, the inherent key escrow of IBE ensures the decryption of blinding factors in necessary cases; for example, when a user wants to decrypt a suspected ticket but fails. Such failures might be caused by malicious operations of the compromised IdP. The IBE private keys are generated by *a trusted coordi-*

nator, and the coordinator is also responsible for in the dispute resolution when a user cannot decrypt suspected tickets. Finally, more efficient search of tickets is designed with Bloom filters, at the expense of some user privacy; otherwise, a user has to try each blindly-signed ticket in the logs one by one (i.e., attempts to decrypt the blinding factor for each ticket entry), when searching for his tickets. In particular, a pseudonym or index of the user, i.e., the result of *his account through a Bloom filter*, is stored along with each of his tickets in the public logs, while other messages of the ticket entry are still kept encrypted or blinded. Then, he is able to quickly filter out most of other users' tickets.

Contributions. We analyze the accountability of IdPs in the SSO services. Then, ticket transparency is proposed to monitor the operations of the IdPs, ensuring that a fraudulent ticket issued by IdPs will be finally detected by the user or the trusted RP. To the best of our knowledge, this is the first attempt to solve the security problems caused by potentially intruded or compromised IdPs in the SSO services.

The remaining of the paper is organized as follows: We present the threat model, security building blocks, and the design of the proposed ticket transparency scheme in Section 2, followed by a security analysis in Section 3 and a performance evaluation in Section 4. Then, we discuss the related work in Section 5 and conclude in Section 6.

2 Accountable Single Sign-On with Privacy-Preserving Public Logs

In this section, we first present the threat model, security goals, and cryptographic building blocks. Then, the designs of ticket transparency are proposed by steps, following the requirements of transparency, privacy and efficiency. Finally, ticket transparency in full view is described.

2.1 Threat Model and Security Goals

Threat Model. A basic SSO scenario includes one IdP, multiple RPs, and a number of users. We assume *the RPs are trusted* since they are the service providers who are responsible to verify the validity of SSO tickets. Ticket transparency aims to prevent unauthorized sign-on to RPs by fraudulent tickets, so the RPs are assumed to be trusted; otherwise, an untrusted RP would allow an attacker to sign on even without any valid tickets. On the contrary, *the online IdP is potentially malicious* and it could be compromised by a number of attacks. The malicious IdP might pretend to be benign and act as expected to avoid being detected, but it could arbitrarily deviate from its specifications at times. For example, the IdP signs fraudulent tickets and then tries its best to conceal their existence by manipulating other messages.

Ticket transparency depends on an independent log server to publicly record SSO tickets against the malicious IdP. As another online system, *the log server*

is assumed to be potentially malicious. It is curious about the users' privacy, so it tries to infer sensitive or private information about the users from the communications and the public logs. Meanwhile, the log server might arbitrarily deviate from its specifications; e.g., not record a ticket in the public logs after signing it. Moreover, the IdP and the log server might collude to sign onto a target RP with fraudulent tickets as a user or leak some private information of the user. They might also collude with some malicious users to take actions. For example, the malicious IdP may register an account in the SSO service, which is completely controlled by itself. Finally, *an offline trusted coordinator in introduced* in ticket transparency. The coordinator acts as the PKG of IBE to generate private keys for all users. It also coordinates the dispute resolution process, when a user suspects there exist a fraudulent ticket labeled with his account but he cannot decrypt this ticket in the public logs.

Security Goals. Ticket transparency cannot eliminate fraudulent tickets, but it ensures that any fraudulent ticket will be detected by the victim user, the RPs and/or the coordinator in the future. In particular, if the IdP issues a ticket without the user's authorization or participation, our scheme ensures: (a) the fraudulent ticket without a valid signature by the log server will not be accepted by the RPs; or (b) the fraudulent ticket with a valid signature by the log server, accepted by the target RP, will be identified from the public logs by the victim user, by comparing the ticket entries with his history of sign-on activities. Here, we assume, when the victim user suspects that an attacker signs on using his account, he remembers his sign-on operations especially in the period during which the suspected sign-on action occurred.

The fraudulent-ticket detection as above requires the log server to record all tickets in the public logs. So our scheme also needs to audit the operations of log servers, to ensure that (a) the log server records all valid tickets in the public logs, and (b) the public logs are append-only. Any log operation deviating from these specifications, will be detected by the RPs. The second goal is to protect users' privacy, as much as possible. In the SSO scenario, a user's privacy is defined as the information about his sign-on activities, such as the account, the requested RP, the occurrence time of sign-on requests, etc. All these privacy data are included in the SSO tickets. To protect users' privacy, we employ blind signatures to hide the contents of SSO tickets in the public logs. However, to check whether a suspected ticket is indeed fraudulent, the coordinator needs to un-blind some relevant ticket entries in the dispute resolution. This process may disclose private information of other users. Overall, ticket transparency ensures the fraudulent-ticket detection at a cost of user privacy – compared to the original SSO protocols [44,50], ticket transparency leaks limited user privacy to the offline trusted coordinator.

2.2 Cryptographic Building Blocks

Ticket transparency is designed on top of three cryptographic building blocks: *blind signature*, *identity-based encryption*, and *Bloom filter*. We define the notations used in this paper and explain each of these building blocks as follows.

- I, L, C, and R_i, denote the IdP, the log server, the coordinator, and the i-th RP, respectively.
- A is the authenticated user, i.e., the victim of the malicious operations by the IdP and/or the log server.

Blind Signature. With a secret factor s, I blinds m into $m' = \mathbb{B}(m, s)$, and L blindly signs m' into $\mathbb{S}_L(m')$. Then, I un-blinds the message signed by L to obtain $\mathbb{S}_L(m) = \mathbb{U}(\mathbb{S}_L(m'), s)$.

Identity-based Encryption (IBE). The trusted coordinator holds the IBE master key and generate the private keys for all users. The IBE public parameters are publicly known, and everyone is able to derive A's IBE public key and encrypt m into $\mathbb{E}_A(m)$, but only A itself and the coordinator can decrypt $\mathbb{E}_A(m)$ into m.

Bloom Filter. We adopt a Bloom filter $\mathbb{F}(\cdot)$ to generate the pseudonym (i.e., index) for A in the form of $N_A = \mathbb{F}(A)$. The Bloom filter is deterministic, with a false negative rate of zero and a false positive rate of α, where $0 < \alpha < 1$. That is, the expectation of the probability $P(U \neq A | N_U = \mathbb{F}(A))$ is α.

2.3 The Ticket Transparency Framework by Steps

Next, we elaborate the design of ticket transparency step by step, following the requirements of *transparency*, *privacy* and *efficiency* of the security-enhanced SSO services.

1. Transparent Ticket. In the original SSO protocols, when attempting to sign onto R_i, user A is redirected from R_i to the IdP, who authenticates A and signs a ticket $m = \mathbb{S}_I(A, R_i, o)$, where o denotes the other necessary information in the ticket specified in the SSO protocols. Then, the IdP responds with the ticket to complete the sign-on process.

Improved Design. Instead of sending m to A, I first sends it to L, which signs the *transparent ticket* in the form of $\mathbb{S}_L(m) = \mathbb{S}_L(\mathbb{S}_I(A, R_i, o))$ and returns it to I. L also records the transparent ticket in the public logs. On receiving $\mathbb{S}_L(m)$, I forwards the ticket to A, who therefore presents the ticket to R_i to complete the sign-on process.

Ticket Verification. After R_i verifies the two signatures on the transparent ticket (and also other verifications as specified in the SSO protocols, including the validity period, the unique identifier against replay attacks, etc.), it accepts the ticket and allows the ticket holder to sign on as A.

Transparency. The process is a straightforward extension of certificate transparency to the SSO scenario. It creates a set of transparent tickets and records them in the public logs. The IdP (or the signing system in an IdP) can no longer issue a valid ticket only by itself, as the ticket has to be signed again by the log server. The tickets in the public logs are always available to enable the fraudulent-ticket detection at any time. Whenever any user suspects that an attacker signing onto R_i on behalf of him using a fraudulent ticket, he will retrieve all tickets under his account from the public logs and detect fraudulent tickets.

2. Blindly-Signed Transparent Ticket. While the basic transparency ticket design defeats against compromised IdPs effectively, it leaks user privacy because all sign-on activities are recorded in the public logs, which are accessible to all users. Therefore, we revise the design to *blindly-signed transparent tickets*. In this scheme, the second signatures are created blindly by the log server, to eliminate the private information in the publicly-visible ticket entries.

Improved Design. After signing a ticket $m = \mathbb{S}_I(A, R_i, o)$, I blinds it into $m' = \mathbb{B}(m, s)$ with a random blinding factor s. I sends m', instead of m, to L along with $\mathbb{E}_A(s)$, which enables A to un-blind the ticket in a later time. Accordingly, L blindly signs m' into $\mathbb{S}_L(m')$ and returns it to I. I un-blinds $\mathbb{S}_L(m')$ into $\mathbb{S}_L(\mathbb{S}_I(m))$ and sends it to A. In the meantime, L creates a ticket entry of $\{\mathbb{S}_L(\mathbb{B}(m, s)), \mathbb{E}_A(s)\}$ and records it chronologically in the public logs.

Privacy and Transparency. First, there is no plaintext information stored in the logs, which protects user privacy from irrelevant entities (i.e., the log server and any other entities without the private key cannot decrypt s). When a user suspects himself is the victim of fraudulent tickets, he can retrieve all ticket entries within the suspected period and applies his private key to decrypt $\mathbb{E}_A(s)$ of every ticket entry. For the tickets issued under his account, he is enabled to recover s correctly and then un-blind the tickets. Therefore, any user can only un-blind his own transparent tickets, but not the ones under others' accounts.

3. IBE-Encrypted Blinding Factor. The above scheme works well, if the compromised IdP is not intelligent or collusive with malicious users. However, the compromised IdP could destroy the fraudulent-ticket detection by issuing fraudulent tickets under A's account but blinding them with the secret blinding factor escrowed to another user Z, who is either malicious or even does not exist. More specifically, a malicious IdP signs a ticket as $m' = \mathbb{B}(\mathbb{S}_I(A, R_i, o), s)$ and sends it along with $\mathbb{E}_Z(s)$ to the log server. This results in a ticket entry $\{\mathbb{S}_L(\mathbb{B}(m, s)), \mathbb{E}_Z(s)\}$ in the public logs. However, when A retrieves this ticket, he cannot decrypt the blinding factor to check whether it is a fraudulent ticket under his account.

We adopt IBE in the framework, utilizing the inherent key escrow of IBE to ensure that the blinding factor can always be decrypted when it is necessary. In our design, the IBE master key is held by the trusted coordinator, and it is always able to decrypt any blinding factor in the public logs. The adoption of IBE also reduces the overhead of key management for the IdP.

Improved Design. When encrypting the blinding factor s into $\mathbb{E}_A(s)$, I derives A's IBE public key and then uses it in the encryption. So A is able to decrypt s by his own IBE private key.

Transparency against the Intelligent IdP. If a user suspects there exist fraudulent tickets labeled with his account, he follows the process described above to detect fraudulent tickets. Then, after trying to decrypt the secret blinding factors of all suspected ticket entries with his IBE private key, if the user still has doubts, he initiates a *dispute resolution* process to the coordinator. Note that to initiate

the process, the user needs to show some reasonable evidences to support his doubt, e.g., an abnormal log in the RP system.

The coordinator C examines the evidences to decide whether the dispute resolution process shall continue and coordinates the process if the evidences are reasonable. In particular, for each suspected ticket entry $\{\mathbb{S}_L(\mathbb{B}(m, s)), \mathbb{E}_A(s)\}$ within the period of dispute, C recovers s with the IBE master key, and then uses s to un-blind the ticket. If and only if s is the correct blinding factor that was used to blind the ticket, C obtains a valid transparent ticket [11]. Then, if this ticket is labeled with the disputer A's account, C sends it to A to allow him to continue the detection as described above; otherwise, if all inquired tickets are associated with other users but not A, C will send nothing to A and the dispute resolution terminates. On the other hand, if it cannot recover a valid transparent ticket using the blinding factor from the IdP, the IdP becomes suspicious – there must be some accidental error or intentional manipulation in the ticket creation.

4. Ticket Entry with Pseudonyms. The ticket entries in the pubic log contain blindly-signed tickets, which provide no explicit information about the user for whom the ticket is issued. Therefore, a user who suspects fraudulent tickets are issued under his account has to try all tickets that are created in the suspected period, which is very inefficient. We adopt a Bloom filter $\mathbb{F}(\cdot)$ to generate pseudonyms for users and store the pseudonyms along with tickets issued under the enclosed accounts in the public logs. As a result, a user and the coordinator only need to un-blind the entries with his pseudonym, in the fraudulent-ticket detection and the dispute resolution.

Improved Design. For each user A, I generates his pseudonym $N_A = \mathbb{F}(A)$, and sends it along with the blinded ticket $\mathbb{B}(\mathbb{S}_I(A, R_i, o), s)$ and the encrypted blinding factor $\mathbb{E}_A(s)$ to L. Accordingly, the ticket entry in the public logs consists of $\{\mathbb{F}(A), \mathbb{S}_L(\mathbb{B}(\mathbb{S}_I(A, R_i, o), s)), \mathbb{E}_A(s)\}$.

Efficiency and Privacy. The pseudonym works as an index to facilitate the ticket search. In particular, in the cases without dispute, a user only needs to check the ticket entries with his pseudonym, instead of all the tickets in the suspicious period. In the cases with dispute, they still need to check all tickets in the suspected period; otherwise, the malicious IdP could send a fake pseudonym to the log server to conceal the existence of fraudulent tickets.

The pseudonyms in ticket entries may leak user privacy, since it provides a way to associate the sign-on activities of a certain user. If an attacker knows the user's account, he can derive the user's sign-on pattern to some extend – the pseudonyms are generated by the Bloom filter, so that *a user is identified but with the false positive rate* of the adopted Bloom filter. Besides, this privacy leakage is limited to the approximate occurrence time of sign-on activities, but without the RP that the user signs onto.

5. Ticket Entry verified by RPs. Two forms of each ticket are processed in the design: (a) *transparent tickets* presented to the RP, i.e., $\mathbb{S}_L(\mathbb{S}_I(A, R_i, o))$; and ($b$) *blindly-signed transparent tickets* recorded in the logs along with the user pseudonyms, i.e., $\mathbb{S}_L(\mathbb{B}(\mathbb{S}_I(A, R_i, o), s))$. As mentioned above, the log server

might not record some tickets in the logs. In this scheme, we require the RP to act as an auditor [32] to audit the operations of the log server. Two forms of each ticket are sent together to the RP, and the blindly-signed transparent tickets enable the audit by RPs against the potentially compromised log server.

Improved Design. The blindly-signed transparent ticket $\mathbb{S}_L(\mathbb{B}(\mathbb{S}_I(A, R_i, o), s))$ and the *plaintext* blinding factor s are sent together by I to A, who forwards them to R_i. Then, R_i un-blinds $\mathbb{S}_L(\mathbb{S}_I(A, R_i, o))$ by itself. Note that these messages are transmitted over secure channels (e.g., HTTPS) as in the original SSO protocols, and s is disclosed to trusted R_i and A only.

Ticket Verification. The RP un-blinds the transparent ticket by itself, and verifies the ticket as the basic transparent ticket design to allow the ticket holder to sign on. Here, it does not verify the relationship between the transparent tickets and the blindly-signed transparent tickets, but un-blinds the blindly-signed ticket by itself to obtain the transparent one instead.

Audit against the Compromised Log Server. With the blindly-signed transparent ticket, the RP further checks whether every valid ticket is recorded in the public logs or not. Similar as certificate transparency [32], the log server builds a Merkle hash tree over all ticket entries. By comparing the root node of the Merkle hash tree and requesting the Merkle audit path for each received valid ticket, the trusted RP checks whether the logs are append-only and the corresponding ticket entry is in the logs or not. The Merkle audit path of an ticket entry, is the shortest list of additional nodes in the Merkle tree to compute the root node [32].

The trusted RP also verifies if the pseudonym stored along with the blindly-signed transparent ticket is consistent with the account enclosed in the transparent ticket. If not, the RP warns the log server about this potential fraudulent ticket and the involved IdP. Therefore, there is no fake pseudonym included in ticket entries.

Privacy in the Dispute Resolution. As mentioned above, due to the detection by the RPs, there is no fake pseudonym in ticket entries. Thus, in the dispute resolution, only the ticket entries including the same pseudonyms as the disputer, instead of all tickets in the suspected period, will be un-blinded by the trusted coordinator.

2.4 Ticket Transparency in Full View

After the above step-by-step analysis, we present ticket transparency in full view.

Initialization Before the SSO service is provided, the IdP generates its key pair to sign tickets, and the log server generates its key pair to blindly sign transparent tickets. Their public keys are publicly known, especially to the RPs. The coordinator initializes the IBE parameters: the IBE public parameters are publicly known, and the IBE master key is held by the coordinator. The Bloom filter is also decided by the SSO service provider. The initial Merkle hash tree of logs is empty, and the root node stored on every RP is initialized as null.

When a user joins, he registers his account in the IdP and applies for his IBE private key from the coordinator. Or, a user may apply for the IBE private

key from the coordinator, only when he suspects any fraudulent tickets labelled with his account and wants to un-blind the suspected tickets.

Sign-On When A attempts to sign onto R_i, he is redirected to I. After authenticating A, I signs a ticket $m = \mathbb{S}_I(A, R_i, o)$, blinds it into $m' = \mathbb{B}(m, s)$ by a random blinding factor s, and computes $N_A = \mathbb{F}(A)$ and $\mathbb{E}_A(s)$. Then, I sends $\{N_A, m', \mathbb{E}_A(s)\}$ to L.

L blindly signs m' into $\mathbb{S}_L(m')$, and sends it to I. An entry of $\{N_A, \mathbb{S}_L(m'), \mathbb{E}_A(s)\}$ is recorded chronologically in the public logs. Then, I sends $\{\mathbb{S}_L(m'), s\}$ to A, and the message is forwarded to R_i. R_i un-blinds $\mathbb{S}_L(m')$ into $\mathbb{S}_L(\mathbb{S}_I(A, R_i, o))$, and verifies the validity of this transparent ticket (including the two signatures on the ticket and other fields). Then, it allows A to sign on if the ticket is valid. It is worthy noting that the communications among I, A and R_i are over secure channels such as HTTPS, as in the original SSO protocol.

Audit R_i regularly requests the root node of the Merkle hash tree from L. It checks whether the logs are append-only; i.e., the root node it stored is a leaf of the current Merkle hash tree. Then, the root node is updated on R_i.

For each received valid ticket $\{\mathbb{S}_L(m'), s\}$, R_i requests the Merkle audit path for $\mathbb{S}_L(m')$, and L responds with the entry of $\{N_A, \mathbb{S}_L(m'), \mathbb{E}_A(s)\}$ and its Merkle audit path to the current root node. R_i checks that it is a valid Merkle audit path for the ticket entry; L is faulty. Next, it checks that $N_A = \mathbb{F}(A)$; otherwise, I is malicious. This audit by R_i may be performed for a batch of tickets. Besides, the messages between I and L are also protected over secure channels, especially with data integrity. If any audit fails, the log server is detected as compromised.

Ticket Search and Dispute Resolution A retrieves all ticket entries where $N_U = \mathbb{F}(A)$, during the interested period. Then, A tries each entry one by one, i.e., attempts to decrypt s to un-blind the transparent ticket. If there is a valid ticket enclosing his account but without his participation, it is detected as a fraudulent ticket.

If A still suspects fraudulent tickets after trying all entries with $\mathbb{F}(A)$ as above, he proposes a dispute with some reasonable evidences supporting this suspect. After C examines this dispute and the evidences, for each suspected ticket entry where $N_U = \mathbb{F}(A)$ but A cannot un-blind the ticket, C decrypts s by the IBE master key and uses it to un-blind the blindly-signed ticket: if the un-blinded result is not a valid transparent ticket, it is asserted that I performed something malicious; otherwise, if A is labelled in this ticket, C sends the transparent ticket to A, which indicates fake $\mathbb{E}_A(s)$ or something abnormal with A's IBE private key; for example, it may result from A's corrupted IBE private key. If it is a valid transparent ticket for another user, nothing is disclosed to A.

3 Security Analysis and Discussion

In this section, we analyze ticket transparency in terms of correctness and privacy. Some extended discussions are presented.

3.1 Correctness

The correctness is proved with the following sketch: (a) An SSO ticket is accepted by RPs, only if both the IdP and the log server have signed it; (b) Every SSO ticket accepted by RPs is recorded in the logs; and (c) All elements of each ticket entry are verified by non-malicious entities (the user, the RPs, or the coordinator), in the ticket creation and logging, or in the dispute resolution. Thus, the IdP and the log server have to follow their specifications; otherwise, any operation deviating from the specifications in the ticket creation or logging, will result in some verification failures. So a user is able to detect fraudulent tickets by comparing the ticket entries with his history of sign-on activities, in the ticket search or in the dispute resolution.

First of all, it is easy to verify that, with our scheme, an SSO ticket is accepted by RPs only if both the IdP and the log server have signed it, because the trusted RPs verify the two signatures on each ticket. Next, every ticket accepted by RPs is recorded in the logs as a blinded message; otherwise, if such a ticket is not recorded in the logs, the log server is detected as compromised by RPs. As described in Section 2.4, the audits by RPs also ensure that, every valid transparent ticket corresponds to a blinded message where $N_A = \mathbb{F}(A)$. Therefore, user A is able to retrieve all his valid tickets in the ticket entries where $N_U = \mathbb{F}(A)$, by his own IBE private key or in the dispute resolution.

If the IdP attempts to prevent the user from identifying such a transparent ticket, it will be detected as malicious in some step. First, the audits by RPs ensure all valid tickets are recorded in the logs. For each valid transparent ticket $\mathbb{S}_L(\mathbb{S}_I(A, R_i, o))$, three elements $\{N_A, \mathbb{S}_L(m'), \mathbb{E}_A(s)\}$ of the corresponding ticket entry are verified by non-malicious entities as follows: (a) the relationship of N_U, $\mathbb{S}_L(m)$, $\mathbb{S}_L(\mathbb{B}(m, s))$ and A is verified by the trusted RP that receives s, when A is signing on; and (b) the relationship of N_U, $\mathbb{S}_L(\mathbb{B}(m, s))$ and $\mathbb{E}_A(s)$ where $N_U = \mathbb{F}(A)$, is verified by A in the ticket search or by the coordinator in the dispute resolution. Specially, a mis-matching blinding factor will not result in the log server's valid signature of $\mathbb{S}_I(A, R_i, o)$ [11]. So the IdP has to encrypt the correct blinding factor in ticket entry, and nobody can tamper with a ticket entry without being detected.

In summary, the IdP cannot conceal the existence of a transparent ticket if it is accepted by RPs. So a ticket issued without the user's participation or authorization, will be detected by the user finally in the ticket search or in the dispute resolution.

3.2 Privacy

First of all, no *extra* user privacy is leaked to the IdP, compared with the original SSO protocols [44,50]. The IdP is always aware of the users' all sign-on activities, either in the original SSO protocols or with ticket transparency.

With ticket transparency, the privacy on a user's sign-on activities is leaked publicly to some extent, compared the original SSO protocols. This privacy leakage is adjustable. Next, we compare different scenarios when the Bloom filter is

adopted or not. In the case of no dispute, if ticket transparency works *without* the Bloom-filtered pseudonyms, there is not any extra privacy leaked, compared with the original SSO protocols. Only blinded or encrypted data are stored in the public logs. However, in the dispute resolution for A, the detailed sign-on activities of all users other than A during the interested period, are disclosed to the trusted coordinator.

On the contrary, after the Bloom filter is adopted to the pseudonyms, some privacy is leaked publicly in the case of no dispute, while the privacy disclosed to the coordinator is relieved. A curious user can learn another user's history of sign-on activities but with a false positive rate (i.e., the false positive rate of $\mathbb{F}(\cdot)$), provided that the attacker knows the victim user's account. Moreover, this privacy leakage is limited to the occurrence of sign-on activities but no information about the RP that is signed onto, because the ticket is blinded. On the other hand, in the dispute resolution, only the sign-on activities of other users who have the same pseudonyms as the disputer, are disclosed to the coordinator.

In summary, the quantity of the user privacy leaked in ticket transparency depends on the adopted Bloom filter. In general, a Bloom filter with a higher false positive rate, protects more user privacy in the case of no dispute but leaks more to the coordinator in the dispute resolution. In the extreme scenario, we may choose the Bloom filter with the maximum false positive rate (i.e., $\alpha = 1$ and it is a constant function), and actually no pseudonym is stored along with the ticket entries. Therefore, user privacy is leaked to nobody except the coordinator in the dispute resolution. So the Bloom filter shall be chosen carefully, because its false positive rate is a trade-off of efficiency and privacy in different cases.

3.3 Discussion

When encrypting the blinding factors, the IdP shall derive a user's IBE public key based on his account and also a period of validity (e.g., the year) [8]. So the user updates his IBE private key periodically. Accordingly, the public logs of ticket entries are organized as multiple Merkle hash trees according to the occurrence time. Then, some trees of logs may be destroyed after a certain number of years, to mitigate the risk due to the unexpected exposure of IBE private keys. Similarly, the logs of certificate transparency are usually organized as multiple logs of SCTs, according to the expiration year of certificates [20].

When implementing the coordinator which is trusted in ticket transparency, distributed PKGs [8,29] and accountable IBE [21,22] can be adopted to generate the IBE private keys for users, to improve the trustworthy of the coordinator.

Customized Bloom filters may be adopted. A user specifies the Bloom filter to generate his pseudonym, so that he is able to decide the false positive rate by himself based on his privacy policy. Such customized design requires the cooperation of the IdP and RPs, when a ticket entry is created and verified. The coordinator also needs to know the customized Bloom filter of each user, in the dispute resolution.

4 Performance Evaluation

We experimentally measured the performance of computationally expensive cryptographic building blocks and present the results as follows. The experiments were conducted with Ubuntu 16.04 on Intel Core i3-6100U CPU (2.30GHz) and 8GB RAM. We estimated the overhead of the proposed scheme at two different levels of security strength [4]. In particular, to achieve 112-bit security, we adopted RSA-2048, Chaum's scheme based on RSA-2048 and Boneh-Franklin scheme based on Elliptic-Curve-224; and for 128-bit security, RSA-3072 and Elliptic-Curve-256 were used. We measured RSA and Chaum's scheme based on `crypto++` [60], and Boneh-Franklin scheme based on the Stanford IBE library [36], which were implemented in C++ and C, respectively.

Table 1. The time cost (in ms) of cryptographic operation.

Cryptographic Operation		Security Strength	
		112-bit	118-bit
Traditional Signature RSA with PKCS1 [48]	Sign	2.0217	10.4863
	Verify	0.0432	0.1099
Blind Signature Chaum's scheme [6, 11]	Blind	0.8899	1.5535
	Sign	2.0456	10.4743
	Un-blind	0.3350	0.8096
	Verify	0.0322	0.0932
IBE BF scheme [8]	Encrypt	11.6635	29.3586
	Decrypt	7.4750	16.1926

For each successful instance of the original SSO protocol, there are only one signing operation by the IdP and one verification of traditional signature by the RP. On the other hand, ticket transparency additionally requires one blinding and one IBE encryption by the IdP, one blind signing by the log server, one un-blinding and one verification of blind signature by the RP. So the public-key cryptographic operations introduced by ticket transparency increase the computation time by about 14.9662 ms with 112-bit security or 42.2892 ms with 128-bit security, according to the experimental results shown in Table 1.

In the original SSO protocol, there are at least three rounds of communications among the IdP, the user and the RP, excluding the authentication of users: (a) the sign-on attempt to the RP, (b) the redirection to the IdP to return the ticket, and (c) the final successful sign-on. Compared with the original protocol, ticket transparency requires only one more round of communications between the IdP and the log server, to blindly sign the ticket and record it in the logs.

We measured the network delay of the Google SSO service, https://accounts.google.com, after a user has been authenticated. Note that, in the real-world deployment, more rounds of communications are possible for one instance of the SSO protocol. In particular, the Google SSO service takes four rounds of communications between the browser and the Google SSO server, after a user

has been authenticated: three to verify and update the cookie in the browser, and the last one to return the ticket (i.e., id_token of OpenID Connect). With the browser running in our laboratory at the University of Kansas, the average delay is 873.49ms, excluding the communications between the browser and RPs. Note that, this delay includes data verifications by the browser and the the Google SSO service. Therefore, we estimate that about 873.49/4 = 218.37ms will be introduced by one round of communications between the IdP and the log server in ticket transparency. Therefore, the total overhead introduced by ticket transparency, in terms of end-to-end processing time, is about 233.34ms with 112-bit security or 260.66ms with 128-bit security, about 25% - 30% of the original SSO protocol.

5 Background and Related Work

5.1 Security of SSO Services

The security of SSO services has been investigated for several years. Implementation vulnerabilities have been discovered in various SSO systems, including OpenID and customized SSO protocols [66], SAML with WS-Security [54], OAuth [57, 63, 64], etc., which allow an attacker to sign onto RPs on behalf of other users or disclose private information of others. Such vulnerabilities are found in widely-used SSO services such as Facebook OAuth [66, 69], Windows LiveID [67] and Google OpenID [34,66]. These works focused on the secure implementation of SSO services, and they cannot address the security problem caused by fraudulent tickets that are issued by compromised IdPs. On the contrary, our work proposes a solution extending the certificate transparency framework, to enhance the accountability of SSO services against the compromised IdPs.

Different anonymous SSO schemes have been proposed to allow users to access RPs without revealing their identities to the RPs, for the global system for mobile communication (GSM) [16], based on broadcast encryption [26], group signatures [65] or extended Chebyshev Chaotic Maps [33]. [25] improves the anonymous SSO schemes, where only the designated RP service is able to verify the tickets, and no identity information is released to the IdP. These approaches also assume trustworthy IdPs, and then cannot handle the problem of fraudulent tickets. The public accountability of the IdPs in anonymous SSO services will be included in our future work.

5.2 CA Security Incidents and Certificate Transparency

Well-known accredited CAs are recently reported to sign fraudulent certificates due to network intrusions [12, 19, 42, 59], reckless identity validations [40, 55, 56, 68], mis-operations [30, 41, 70], or government compulsions [15, 53]. Lessons are learned from these security incidents in an unexpected way that a signing system which is usually built on HSMs in well-protected organizations, could still be compromised to sign fraudulent messages, which contain well-formatted and verifiable but incorrect or misleading data [12, 38, 59].

Certificate transparency has been widely adopted in the Internet. [1,17,24,45] investigated the deployment of certificate transparency from different aspects. CASTLE [38] attempts to eliminate fraudulent certificates, by designing an air-gapped and completely-touchless signing system for CAs. However, it is designed only for low-volume certificate workloads and then does not work for online services such as SSO. Threshold signature schemes [13, 51] are also applied in CAs to protect the private key to sign certificates. Distributing the private key magnifies the difficulty to compromise the confidentiality of the key [28, 46], but multiple administrators are required to check whether the well-formatted to-be-signed request includes correct information or not [38], which is impractical in online services.

The principles of certificate transparency have been extended in different ways. Revocation transparency [31] extended the approach to provide publicly-accountable certificate revocation in PKIs. CIRT [49] achieved both certificate transparency and revocation transparency by recording certificates in two Merkle hash trees – one of which is in chronological order with all certificates and the other is lexicographical with only the recent ones for each certificate subject. Then, Singh et al. [52] improved CIRT by bilinear-map accumulators and binary trees, to achieve the transparencies with shorter proofs. CONIKS [39] presented transparent key directories based on Merkle prefix trees, allowing its users to audit their public keys while preserving the user privacy. Insynd [47] proposed privacy-preserving transparent logging by authenticated data structures, so its users are enabled to take actions without disclosing everything in the log server.

On top of certificate transparency, binary transparency [43] ensures that all versions of Firefox binaries are publicly logged. When a version is released, a Merkle hash tree is computed over all binaries, and the root node of the hash tree is bound in a certificate that is transparent in public logs. Attestation transparency [5] integrates certificate transparency with Intel Software Guard eXtension (SGX) so that a legacy client can verify the service by establishing a TLS channel with the service's private key that is protected in SGX enclaves, while the corresponding certificate is transparently logged. Inspired by the design of CT, the general transparency overlay [10] is proposed, which can be instantiated to implement transparency for other services.

The certificate transparency framework and its variations provide good references to solve the problem of fraudulent SSO tickets. However, these solutions cannot be directly adopted due to the privacy concerns that are inherent in SSO tickets. Therefore, customized privacy-preserving techniques are necessary to be integrated into the design of ticket transparency.

5.3 Accountability of Third-Party Services

The public accountability of cloud services is one of the important topics of cloud security. Third-party auditors are introduced to check the integrity of outsourced data in untrusted cloud systems, while the data contents are not leaked to the auditors [61, 62, 62]. Liu et al. present consistency as a service [35]: a data cloud is maintained by the cloud service provider, and a group of

users that constitute an audit cloud verify whether the data cloud provides the promised level of consistency. PDP [2] and POR [9] enable the remote tenants to verify whether the data are intact in the untrusted clouds, with lightweight complexity of communications and computations. [27] detects CPU cheating on virtual machines maintained by semi-trusted cloud providers, using CPU-intensive calculations. The SSO services can be viewed as another kind of cloud services – ID as a service, but the accountability of untrusted IdPs have not been well investigated in the literature.

The private key generator (PKG) of IBE is responsible for generating the private keys for all users, so that the PKG has to be fully trusted by all users. This problem of inherent key escrow is mitigated by different ways: (a) distributed PKGs [8,29], where the IBE master key is distributed among several independent components, or (b) accountable IBE [21,22] – if the PKG regenerates the private key for any user, a proof will be produced automatically.

We share the same spirit with these solutions that the trust of a third-party service need to be reduced. Meanwhile, distributed PKGs and accountable IBE can be adopted in our scheme to generate the IBE private keys for users, to improve the trustworthy of the coordinator; see Section 3.2 for details.

6 Conclusion and Future Work

The SSO services have become more and more popular in the Internet. An SSO ticket issued by an IdP allows a user to sign onto numerous RPs on behalf of the account labeled in the ticket. However, recent security incidents of accredited commercial CAs indicate that a well-protected and fully-trusted signing system might be compromised to sign fraudulent messages. Therefore, we argue that the online signing system of SSO services will become an attractive target of interests to adversaries so that they should not be fully trusted. So we need a new security mechanism to detect and/or prevent fraudulent SSO tickets signed by potentially compromised IdPs.

This paper proposes *ticket transparency*, the first open framework in the literature to provide the public accountability of IdPs. With ticket transparency, each IdP-signed ticket is recorded in the public logs, so that any user who suspects there are fraudulent tickets issued under his account is enabled to detect fraudulent tickets in the logs. To achieve ticket transparency while mitigate the privacy leakage by the utilization of public log, we integrate blind signatures, IBE, and Bloom filters in the design to balance transparency, privacy and efficiency in different scenarios. The preliminary performance analysis shows that ticket transparency introduces a reasonably small overhead in the SSO services.

In the future, we plan to prove the security of ticket transparency in a more formal way as [14]. We will also build the prototype on top of open-source SSO systems, to finish more experiments.

References

1. J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz. Mission accomplished? HTTPS security after DigiNotar. In *17th Internet Measurement Conference (IMC)*, pages 325–340, 2017.
2. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Perterson, and D. Song. Provable data possession at untrusted stores. In *14th ACM Conference on Computer and Communication Security (CCS)*, pages 598–610, 2007.
3. Guangdong Bai, Jike Lei, Guozhu Meng, Sai Sathyanarayan Venkatraman, Prateek Saxena, Jun Sun, Yang Liu, and Jin Song Dong. Authscan: Automatic extraction of web authentication protocols from implementations. In *NDSS*, 2013.
4. E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. SP 800-57 - Recommendation for key management - Part 1: General. Technical report, National Institute of Standards and Technology, 2006.
5. J. Beekman, J. Manferdelli, and D. Wagner. Attestation transparency: Building secure Internet services for legacy clients. In *11th ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 687–698, 2016.
6. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.
7. B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
8. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - Crypto*, pages 213–229, 2001.
9. K. Bowers, A. Juels, and A. Oprea. Proofs of retrievability: Theory and implementation. In *ACM Workshop on Cloud Computing Security (CCSW)*, pages 43–54, 2009.
10. Melissa Chase and Sarah Meiklejohn. Transparency overlays and applications. In *13th ACM Conference on Computer and Communications Security (CCS)*, pages 168–179, 2016.
11. D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology - Crypto*, pages 199–203, 1982.
12. Comodo Group Inc. Comodo report of incident, 2011. https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html.
13. Y. Desmedt. Society and group oriented cryptography: A new concept. In *Advances in Cryptology - Crypto*, pages 120–127, 1987.
14. Benjamin Dowling, Felix Günther, Udyani Herath, and Douglas Stebila. Secure logging schemes and certificate transparency. In *21st European Symposium on Research in Computer Security (ESORICS)*, pages 140–158, 2016.
15. P. Eckersley. A Syrian man-in-the-middle attack against Facebook, 2011. https://www.eff.org/deeplinks/2011/05/syrian-man-middle-against-facebook.
16. K. Elmufti, D. Weerasinghe, M. Rajarajan, and V. Rakocevic. Anonymous authentication for mobile single sign-on to protect user privacy. *International Journal of Mobile Communications*, 6(6):760–769, 2008.
17. O. Gasser, B. Hof, M. Helm, M. Korczynski, R. Holz, and G. Carle. In log we trust: Revealing poor security practices with certificate transparency logs and internet measurements. In *19th International Conference on Passive and Active Measurement (PAM)*, pages 173–185, 2018.
18. Mohammad Ghasemisharif, Amrutha Ramesh, Stephen Checkoway, Chris Kanich, and Jason Polakis. O single sign-off, where art thou? an empirical

analysis of single sign-on account hijacking and session management on the web. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1475–1492, Baltimore, MD, 2018. USENIX Association.
19. GlobalSign. Security incident report, 2011. https://www.globalsign.com/resources/globalsign-security-incident-report.pdf.
20. Google Inc. Known logs, 2018. http://www.certificate-transparency.org/known-logs.
21. V. Goyal. Reducing trust in the PKG in identity-based cryptosystems. In *Advances in Cryptology - Crypto*, pages 430–447, 2007.
22. V. Goyal, S. Lu, A. Sahai, and B. Waters. Black-box accountable authority identity-based encryption. In *15th ACM Conference on Computer and Communications Security (CCS)*, pages 427–436, 2008.
23. M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. Nielsen, A. Karmarkar, and Y. Lafon. W3C recommendation - SOAP version 1.2 part 1: Messaging framework (2nd edition), 2007.
24. J. Gustafsson, G. Overier, M. Arlitt, and N. Carlsson. A first look at the CT landscape: Certificate transparency logs in practice. In *18th International Conference on Passive and Active Measurement (PAM)*, pages 87–99, 2017.
25. J. Han, L. Chen, S. Schneider, H. Treharne, and S. Wesemeyer. Anonymous single-sign-on for n designated services with traceability. In *23rd European Symposium on Research in Computer Security (ESORICS)*, pages 470–490, 2018.
26. J. Han, Y. Mu, W. Susilo, and J. Yan. Anonymous single-sign-on for n designated services with traceability. In *6th International Conference on Security and Privacy in Communication Networks (SecureComm)*, pages 181–198, 2010.
27. R. Houlihan, X. Du, C.-C. Tan, J. Wu, and M. Guizani. Auditing cloud service level agreement on VM CPU speed. In *IEEE International Conference on Communications (ICC)*, pages 799–803, 2014.
28. Jiwu Jing, Peng Liu, Dengguo Feng, Ji Xiang, Neng Gao, and Jingqiang Lin. ARECA: A highly attack resilient certification authority. In *1st ACM Workshop on Survivable and Self-Regenerative Systems (SSRS)*, pages 53–63, 2003.
29. A. Kate and I. Goldberg. Distributed private-key generators for identity-based cryptography. In *7th International Conference on Security and Cryptography for Networks (SCN)*, pages 436–453, 2010.
30. A. Langley. Further improving digital certificate security, 2013. https://security.googleblog.com/2013/12/further-improving-digital-certificate.html.
31. B. Laurie and E. Kasper. Revocation transparency, 2012. http://sump2.links.org/files/RevocationTransparency.pdf.
32. B. Laurie, A. Langley, and E. Kasper. IETF RFC 6962 - Certificate transparency, 2014.
33. T.-F. Lee. Provably secure anonymous single-sign-on authentication mechanisms using extended Chebyshev Chaotic Maps for distributed computer networks. *IEEE Systems Journal*, 12(2):1499–1505, 2018.
34. W. Li and C. Mitchell. Analysing the security of Google's implementation of OpenID Connect. In *13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pages 357–376, 2016.
35. Q. Liu, G. Wang, and J. Wu. Consistency as a service: Auditing cloud consistency. *IEEE Transactions on Network and Service Management*, 11(1):25–35, 2014.
36. B. Lynn. Stanford IBE library v0.7.2. https://github.com/SEI-TTG/id-based-encryption.

37. Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. Do not trust me: Using malicious idps for analyzing and attacking single sign-on. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 321–336. IEEE, 2016.
38. S. Matsumoto, S. Steffen, and A. Perrig. CASTLE: CA signing in a touch-less environment. In *32nd Annual Computer Security Applications Conference (ACSAC)*, pages 546–557, 2016.
39. M. Melara, A. Blankstein, J. Bonneau, E. Felten, and M. Freedman. CONIKS: Bringing key transparency to end users. In *24th USENIX Security Symposium*, pages 383–398, 2015.
40. Microsoft. MS01-017: Erroneous VeriSign-issued digital certificates pose spoofing hazard, 2001. https://technet.microsoft.com/library/security/ms01-017.
41. B. Morton. Public announcements concerning the security advisory, 2013. https://www.entrust.com/turktrust-unauthorized-ca-certificates.
42. B. Morton. More Google fraudulent certificates, 2014. https://www.entrust.com/google-fraudulent-certificates/.
43. Mozilla. Binary transparency, 2017. https://wiki.mozilla.org/Security/Binary_Transparency.
44. A. Nadalin, C. Kaler, R. Monzillo, and P. Hallam-Baker. OASIS standard - Web services security: SOAP message security 1.1, 2006.
45. C. Nykvist, L. Sjostrom, J. Gustafsson, and N. Carlsson. Server-side adoption of certificate transparency. In *19th International Conference on Passive and Active Measurement (PAM)*, pages 186–199, 2018.
46. Erman Pattuk, Murat Kantarcioglu, Zhiqiang Lin, and Huseyin Ulusoy. Preventing cryptographic key leakage in cloud virtual machines. In *23rd USENIX Security Symposium*, 2014.
47. R. Peeters and T. Pulls. Insynd: Improved privacy-preserving transparency logging. In *21st European Symposium on Research in Computer Security (ESORICS)*, pages 121–139, 2016.
48. RSA Laboratories. PKCS #1 v2.2: RSA cryptography standard. Technical report, EMC Corporation, 2012.
49. M. Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *21st ISOC Network and Distributed System Security Symposium (NDSS)*, 2014.
50. N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and M. Chuck. OpenID Connect Core 1.0, 2014. http://openid.net/specs/openid-connect-core-1_0.html.
51. V. Shoup. Practical threshold signatures. In *Advances in Cryptology - EuroCrypt*, pages 207–220, 2000.
52. Abhishek Singh, Binanda Sengupta, and Sushmita Ruj. Certificate transparency with enhancements and short proofs. In *22nd Australasian Conference on Information Security and Privacy (ACISP)*, pages 381–389, 2017.
53. C. Soghoian and S. Stamm. Certified lies: Detecting and defeating government interception attacks against SSL. In *15th Financial Cryptography and Data Security Conference (FC)*, pages 250–259, 2012.
54. J. Somorovsky, A. Mayer, J. Schwenk, M. Kampmann, and M. Jensen. On breaking SAML: Be whoever you want to be. In *21st USENIX Security Symposium*, pages 397–412, 2012.
55. SSL Shopper. SSL certificate for mozilla.com issued without validation, 2008. https://www.sslshopper.com/article-ssl-certificate-for-mozilla.com-issued-without-validation.html.

56. Start Commercial (StartCom) Limited. Critical event report, 2008. https://blog.startcom.org/wp-content/uploads/2009/01/ciritical-event-report-12-20-2008.pdf.
57. S.-T. Sun and K. Beznosov. The devil is in the (implementation) details: An empirical analysis of OAuth SSO systems. In *19th ACM Conference on Computer and Communications Security (CCS)*, pages 378–390, 2012.
58. San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey, and Konstantin Beznosov. What makes users refuse web single sign-on?: an empirical investigation of openid. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 4. ACM, 2011.
59. VASCO Data Security International Inc. DigiNotar reports security incident, 2011. https://www.vasco.com/about-vasco/press/2011/news_diginotar_reports_security_incident.html.
60. J. Walton and the Crypto++ community. Crypto++ library 7.0. https://cryptopp.com/.
61. C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for secure cloud storage. *IEEE Transactions on Computers*, 62(2):362–375, 2013.
62. C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM*, pages 525–533, 2010.
63. H. Wang, Y. Zhang, J. Li, and D. Gu. The Achilles heel of OAuth: A multi-platform study of OAuth-based authentication. In *32nd Annual Computer Security Applications Conference (ACSAC)*, pages 167–176, 2016.
64. H. Wang, Y. Zhang, J. Li, H. Liu, W. Yang, B. Li, and D. Gu. Vulnerability assessment of OAuth implementations in Android applications. In *31st Annual Computer Security Applications Conference (ACSAC)*, pages 61–70, 2015.
65. J. Wang, G. Wang, and W. Susilo. Anonymous single sign-on schemes transformed from group signatures. In *5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pages 560–567, 2013.
66. R. Wang, S. Chen, and X. Wang. Signing me onto your accounts through Facebook and Google: A traffic-guided security study of commercially deployed single-sign-on web services. In *33rd IEEE Symposium on Security and Privacy (S&P)*, pages 365–379, 2012.
67. R. Wang, Y. Zhou, S. Chen, S. Qadeer, D. Evans, and Y. Gurevich. Explicating SDKs: Uncovering assumptions underlying secure authentication and authorization. In *22nd USENIX Security Symposium*, pages 399–314, 2013.
68. K. Wilson. Distrusting new CNNIC certificates, 2015. https://blog.mozilla.org/security/2015/04/02/distrusting-new-cnnic-certificates/.
69. Y. Zhou and D. Evans. SSOScan: Automated testing of web applications for single sign-on vulnerabilities. In *23rd USENIX Security Symposium*, pages 495–510, 2014.
70. M. Zusman. Criminal charges are not pursued: Hacking PKI, 2009. https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-zusman-hacking_pki.pdf.