Methods and Algorithms for Detecting Compromise of Secret Keys

S. Kuzmicheva, M. Kiryakina and S. Zapechnikov

Abstract It is not uncommon for a single user to use multiple devices to access applications, such as messengers. The legitimacy of using client devices and access rights should be monitored to ensure the necessary level of information security. In the paper authors analyze and discuss some approaches to detect unauthorized usage of cryptographic keys. The first approach is known as trace-independent inconsistency, the second approach is an observation of contradiction, the third approach called an observation of acausality. All three approaches are based on the additional dedicated audit service to the application under protection. The main result of our investigation is the technique for applying these principles to the secure communication and messaging protocols. In the paper authors suggest a cryptographically protocol for detecting compromise of secret keys based on blockchain technology.

Keywords Protocol · Trace-independent inconsistency · Observing contradictions · Observing acausality · Detecting compromise of secret keys · Protocol for detecting compromise of secret keys

1 Introduction

Information technologies have reached an important place in the modern world. Different methods of searching, collecting, storing, processing, providing, distributing information are improving. The information became the key asset in our life.

S. Kuzmicheva (✉) · M. Kiryakina · S. Zapechnikov
National Research Nuclear University MEPhI, Moscow, Russia
e-mail: tz7sveta@yandex.ru

M. Kiryakina
e-mail: m.kiryakina@gmail.com

S. Zapechnikov
e-mail: SVZapechnikov@mephi.ru

S. Zapechnikov
Research Center for Cryptocurrencies and Digital Assets, Moscow, Russia

However, rapid development is observing in the methods of illegitimate handling of information. The attackers are using all new mechanisms in attacking.

Specialists in the field of information security are working on important areas that can qualitatively improve the rates of prevention and detection of computer crimes. It's important to detect an attacker at the early stages of an attack, as well as the detection of internal attackers. If an attacker has received trusted access to the environment, it is extremely difficult to distinguish his actions from those of a regular user. Internal violators have a peculiarity: since they have trusted access to the entire environment, traditional solutions for information security cannot handle such cases. Therefore, it is necessary to raise a question about developing an approach of detecting unauthorized user actions with keys information by monitoring using client devices and access rights.

2 Directions of Protection Against Compromise of Cryptographic Keys

It is possible to distinguish two directions in the area of protection against the compromise of cryptographic keys [1]:

1. Keys copy/clone protection

Examples:

- Protection implemented in Common Access Card (CAC)

This point means using a corporate card issued to an employee. That provides to the employee an access to buildings, company data or company objects. In this context, CAC means a set of different access cards.

- Using Certification Authority (CA)

The certificate authority is part of the Public Key Infrastructure in conjunction with the Registration Authority (RA), which verifies the information provided by the certificate requester. If the information was verified, the CA would issue a certificate. Certificates are using for securely work on a network, electronic document management mechanisms, etc.

- Using Keyless SSL technology

The CDN provider Cloudflare can establish secure connections on behalf of its clients without having their secret keys in hand [2].

Keyless SSL allows sites to use Cloudflare's SSL service while maintaining local storage of their private keys. Cloudflare SSL Standard Service requires the client to share their site's SSL key with Cloudflare [2].

Cloudflare modifies the traditional handshake when setting up an SSL connection so that a key server is used instead of the secret key. It simply returns the "secret" value that is signed by the secret key. At the same time, the key server is located on the client's local network, and Cloudflare does not have access to it [2].

2. Protection against unauthorized use of a copy of the key [1].

Let's imagine a situation where an attacker gained access to secret keys and used this information. Most often, a compromised account is using to log into the service, request some documents, etc. It is possible to identify several areas of control to determine the "suspicious" using of credentials [1]:

1. Storing the information of the last visit of the user ("last logon") [1]. It allows you to notify the user about the details of his recent connections;
2. Using Certificate Transparency mechanisms (public audit of using the issued certificates) [1]. It allows you to determine the misuse of key information, however, the decision about the legitimacy of using the information is not making by algorithms;
3. Determining compromise on the protocol level [1, 3]. It allows you to detect a compromise and automatically initiate the process of revoking compromised keys in the system.

3 Approaches to Detecting Compromise on the Protocol Level

There are three types of observing illegitimate interaction of agents within the network [1]. The first one is trace-independent inconsistency, in this way each individual message in the set does not require information about previous events [1]. The second one is an observation of contradiction, it refers to the observation of a sequence of messages that could not occur in this sequence without compromises [1]. The third approach is observation of acausality, which is a sequence of received messages, that requires an assessment of the contradictions of the available knowledge of the agent [1].

3.1 Trace-Independent Inconsistency

Receiving a message that could not occur in any trace of the considered message set is one of the easiest and most convenient way to recognize that the current message sequence is incompatible with the previous ones [1].

To illustrate this method, we can simulate the situation Fig. 1, when Alice has a secret *kA* which she should use to confirm her identity and send a message to Bob. Also, we have Eva, Eva is an attacker, who compromises this secret and sends an authenticated message.

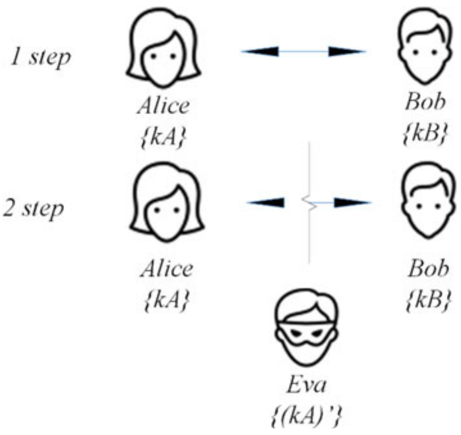

Fig. 1. Example of a trace-independent inconsistency

This type of detection of abnormal behavior indicates that the received message does not match the set of previous messages, regardless of the current trace. Therefore, this method is also very convenient, since we consider the incident from the point of view of "trace-independent". However, it is worth noting that this type of observation relates to theoretical knowledge. In practice, the attacker does not allow such blunders [1].

3.2 Observing Contradictions

This method requires some knowledge about the previous messages to determine the presence of conflicting messages in newly observed messages [1]. Any message can occur individually, but it cannot occur in combination with the previous messages.

For example, we can imagine a dialog between Alice and Bob, it is illustrated in the image Fig. 2. For communication they should have signing keys *kA* and *kB* respectively, and send each other messages, which are tested with their keys over a public channel. Each of them has a counter (which increments, when one of them sends a message) and it also generates a new nonce (*nA* or *nB*). Each message includes the current counter value, a newly generated nonce (*nA* or *nB*) and the last received value nonce of the opponent.

When Alice sends a message to Bob her counter value increments, generates a new nonce *nA* and includes them both in her message along with the last nonce received from Bob. When Bob receives this message, he checks if the counter values do not match, increments his counter, and checks to see if the current counter value, passed by Alice matches.

Sending a message from Bob to Alice is similar, except that Bob sends his new value of nonce *nB*. In this situation, it would be very easy to detect key compromise, if the attacker sent a message from the key owner, because the values of the counters

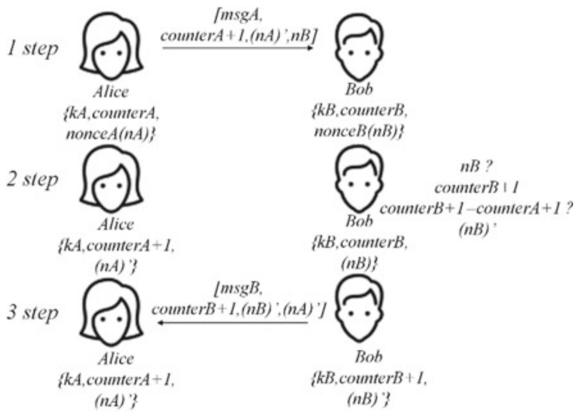
1 step
[msgA, counterA+1,(nA)',nB]
Alice {kA,counterA, nonceA(nA)}
Bob {kB,counterB, nonceB(nB)}
2 step
Alice {kA,counterA+1, (nA)'}
Bob {kB,counterB, (nB)}
nB ? counterB+1 counterB+1=counterA+1 ? (nB)'
3 step
[msgB, counterB+1,(nB)',(nA)']
Alice {kA,counterA+1, (nA)'}
Bob {kB,counterB+1, (nB)'}

Fig. 2. Example of a observing contradictions

will not match. But the main disadvantage of this method is the situation when the attacker compromises both key values. Then they will be able to send false messages to opponents, and then change the values of their counters to the same [1].

3.3 *Observing Acausality*

This method is based on the observation of messages that contradict the agents' knowledge of their activity [1]. This is only valid for agents that causally precede the observed sequence.

To illustrate this method, consider the dialogue between Alice and Bob, it is illustrated in the image Fig. 3. In each of her messages, Alice includes a new value of nonce *nA* and a hash chain of the previous nonces already used in the conversation

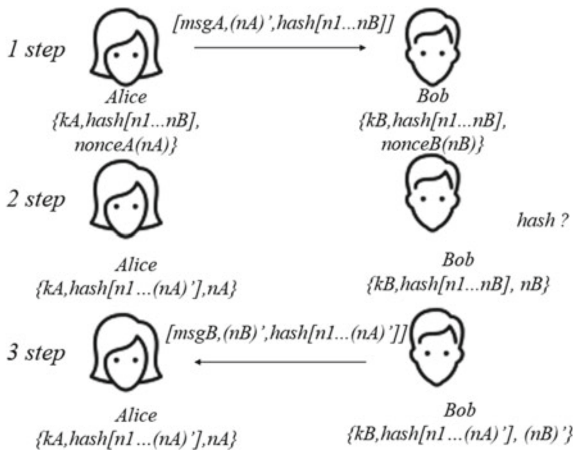
1 step
[msgA,(nA)',hash[n1...nB]]
Alice {kA,hash[n1...nB], nonceA(nA)}
Bob {kB,hash[n1...nB], nonceB(nB)}
2 step
hash ?
Alice {kA,hash[n1...(nA)'],nA}
Bob {kB,hash[n1...nB], nB}
3 step
[msgB,(nB)',hash[n1...(nA)']]
Alice {kA,hash[n1...(nA)'],nA}
Bob {kB,hash[n1...(nA)'], (nB)'}

Fig. 3. Example of a observing acausaliti

by both parties. When the message is received, Bob checks the value of the nonces hash chain with his own and if they match that, he includes the new value *nA*. And when he sends a message to Alice, Bob includes a new nonce *nB* and an updates hash chain. Assume that the attacker Eva compromises Alice's key *kA*, the current nonce value, and the hash chain. Then Eva can start a conversation with Bob and it will naturally increase the hash chain. But in that moment, when Eva emerges from the dialogue of Alice and Bob, then the surfing chat session would be detected, since the hash chains which Alice and Bob have will not be the same [1].

This method of using the hash chain is indicated as promising for the future using [4]. However, there is a problem of assessing the increase in the amount of data transmitted using hash chains. Another problem of this way is assessing the load on the hardware component of the interaction participants. It should also be noted that it is necessary to study the possibility of using lightweight versions of such algorithms applicable to mobile devices. One of such opportunities is a blockchain technology [4].

4 Protocol for Detecting Compromise of Cryptographic Keys Based on Blockchain Technology

The study proposed a protocol for detecting a compromise of private keys based on the audit of user and system actions using enhanced authentication.

The protocol is based on blockchain, which stores transmitted messages. Authentication by user ID, who can write transmitted messages to the blockchain, is also added.

Data exchange between users takes place through the blockchain. Each user has a running agent that checks the message queue for that user in real time (that is, all messages pass through the blockchain). Key distribution is based on Public Key Infrastructure (PKI).

The protocol is applicable for systems with additional authentication and verification of the integrity of the transmitted information checking.

This concept also can be applied to ensure the security of mobile applications in centralized system architecture.

The role mechanism of the presented protocol is following:

Client component:

- Message initiator—the role of the device that sends the message
- The recipient of the message—the role of the device that conducts the reception of the message

Server component:

- Server of storage the information—the server component, carrying storage transmitted information based on blockchain.

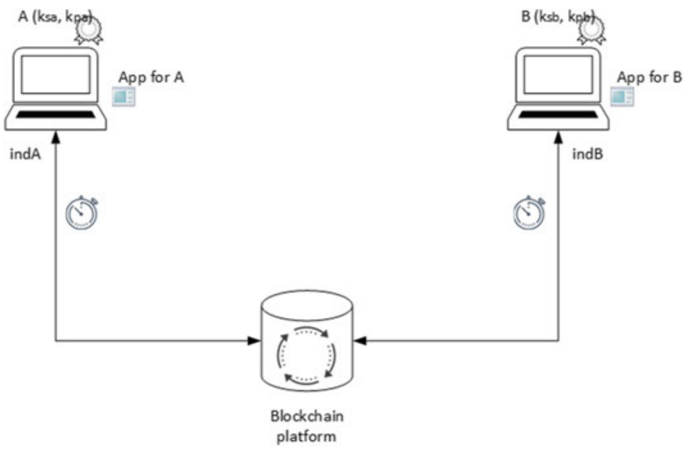

Fig. 4. Schema of the protocol for detecting compromise based on blockchain technology

- Communication server—the role of the server component involved in the processing of message queues for the exchange of information between users.
- Certification authority server—the role of the server component, combining the role of certification authority and registration authority.

For different purposes, one server component may combine the role of storage server, communication server and certification authority server on the same device.

For the schema on Fig. 4 we define the role of the components in the following way:

- *A*—initiator
- *B*—recipient
- Serv—the server combining server roles as storage of information, communication server, certification authority server with key pair *cert(skS, pkS).*

4.1 Working with Keys: Primary Initialization

1. Initiator *A* generates a key pair *skA* and *pkA*
2. *A* receives a certificate for a key pair in the certificate authority *Serv cert(skA, pkA)*
3. The recipient *B* generates a key pair *skB* and *pkB*
4. *B* receives the certificate for the key pair with a certification authority *Serv cert(skB, pkB)*
5. The installed on *A* device agent application generates an *indA* based unique hardware characteristics of the initiator

6. The installed on *B* device agent application generates an *indB* based on the unique hardware characteristics of the recipient
7. And sends the encrypted on *pkS* the ID of *indA* to server *Serv*
8. The server performs the reverse conversion using *skS* and obtains *indA*
9. The server performs a hashing operation on *indA* and stores the reference identifiers of the interaction participants in its database
10. Steps 7–9 are performed for the *B*.

4.2 Sending and Receiving Processes

Sending:

1. *A* sends a message *msg* to *B* containing: *[(msg)pkB, [hash(msg-1), hash(indA)]pkS]*, where *msg-1* is the previous message.
2. The server *Serv* receives the message from *A*, decrypts part of the message using *skS*, compares the hash values *(msg-1)* with the value stored in the blockchain. This step ensures that the message flow between the principal and the server is not disrupted.
3. The server compares the *hash(indA)* received in the message and the reference hash stored in its own database. The server authenticates the initiator.
4. The server calculates the *hash(msg)* and writes to the blockchain.
5. The server updates the status of the message queue for *B*.

Reception:

1. The agent application on device *B*, listening to the message queue for *B* on the *Serv* server, receives a new message.
2. *B* is using *skB* and receives the original *msg* message.

4.3 Principle of Attack Detection

The protocol allows detecting the compromise of key information in the following cases:

- blockchain message sequence mismatch
- inability to authenticate the participant by the identifier (including the situation when it is impossible to confirm the session on the master device).

5 Conclusion

In this paper, existing methods of monitoring anomalies in the agent's communications within the network and their working with cryptographic keys were analyzed.

Cryptographically protocol for detecting compromise of secret keys based on blockchain technology was developed as a goal of this work.

Possible further directions for investigation of secret keys compromise detecting techniques are:

- analysis of safety offered protocol by building a model in Tamarin prover tool;
- proof of concept based on the developed application;
- testing and analyzing of developed protocol.

References

1. Milner, K., Cremers, C., Yu, J., Ryan, M.: Automatically detecting the misuse of secrets: foundations, design principles, and applications (2017)
2. Overview of Keyless SSL: https://www.cloudflare.com/ssl/keyless-ssl/. Accessed 15 Dec 2018
3. Yu, J., Ryan, M., Cremers, C.: DECIM: detecting endpoint compromise in messaging (2018)
4. Zapechnikov, S., Kuzmicheva, S., Kiryakina, M: Classification of methods for detecting compromised private keys (In Russian). In: Materials of the conference Inforino, pp. 110–113 (2018)