Using the blockchain to add automated financial incentives to the Public Key Infrastructure

Justus Fries
Advisor: Heiko Niedermayer
Seminar Innovative Internet-Technologien und Mobilkommunikation SS2017
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: fries@in.tum.de

ABSTRACT

The Transport Layer Security Protocol in its current implementation is based on a centralized and intransparent infrastructure. These flaws have been the cause of Man-in-the-middle (MitM) attacks, which are most commonly rooted in compromised Certificate Authorities. Log-based enhancements, such as Certificate Transparency, have made an effort to solve these problems by logging every signed certificate and thus making signing a public process. However these systems lack proper financial incentives and automation. In this seminar paper Instant Karma PKI (IKP) [15], a system to improve on Log-based enhancements and on the Public Key Infrastructure (PKI), is described and discussed.

Keywords

IKP,PKI,certificate,transparency,blockchain,incentives

1. INTRODUCTION

Secure data transfer on the internet is based on the Transport Layer Security protocol (TLS) [8]. It allows two actors on the web to communicate in privacy and provides data integrity. The most commonly used protocol based on TLS is HTTPS [18], which allows transfer between client and server in an end-to-end encrypted connection.

The basis for the trust model in TLS is the Public Key Infrastructure (PKI). The PKI has two main actors, the websites that want to establish trust in their public key for secure communication, and the Certificate Authorities (CA), who sell a certification of those public keys. The trust in those certificates is established via a chain of trust, which is rooted in certificates bundled with web browsers and operating systems. An example for this can be seen in figure 1. The main point of failure in this system are the CAs themselves. Worldwide, CAs have been compromised and used for Man-in-the-middle attacks. Examples for this are TURKTRUST [10] from Turkey, VeriSign [16, 23] and Comodo [19] from the U.S., DigiNotar [2] and GlobalSign [24] from the Netherlands. One of the newest incidents came from Symantec [21], one of the largest CAs. The issue with these compromises is that every single CA can issue certificates for any website and thus compromise even big websites like Google or Facebook. This problem is facilitated by the fact that browsers contain hundreds of root certificates.

One approach to solve this, proposed by Google themselves, called Certificate Transparency (CT) was published as an experimental RFC in 2013 [14]. It is a log-based approach that is supposed to be publicly monitored, thus making detection of unauthorized certificates faster and easier.

The authors of IKP [15] identified two main goals with their system. It should add incentives for all actors to behave correctly and everything should be automated. This will be described in more detail in Section 3.1. In Section 2 background information to understand IKP is presented, Section 3 deals with the components of the Instant Karma PKI and evaluates its functionality.

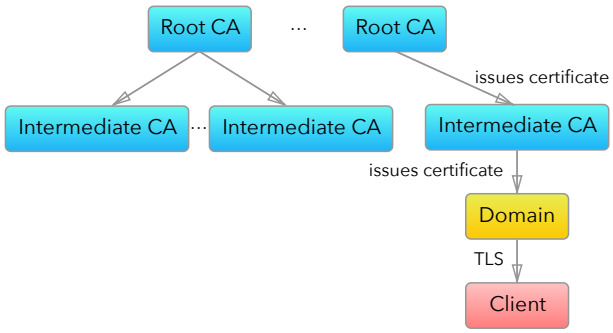

Figure 1: Chain of trust in the public key infrastructure.

2. BACKGROUND

This section describes background information that is important to understand the IKP architecture and why its implementation is desirable.

2.1 Log-based enhancements

Log-based enhancements such as Certificate Transparency [1] aim to move the process of signing certificates to the public and thus make detecting malicious certficates faster by keeping a log of every certificate signed. This log has certain cryptographic features, for example being append-only, that allow verification of the logs themselves and of the existence of a certificate in the log. Furthermore certain important characteristics such as invulnerability to malicious actors [9] can be proved. In the CT system the browsers leverage these features to extend the TLS protocol to include verification that a certificate has been logged using the Signed Certificate Timestamp extension to X.509 certificates [7], this verification of logging is part of the extended validation process. Log operators are for example Google or CAs. Other important actors are the monitors who watch for suspicious

doi: 10.2313/NET-2017-09-1_05

certificates in the logs and reporting those. They also make sure that all logged certificates are visible by periodically fetching new entries and exchanging this information with other monitors and auditors using a gossip protocol.

2.2 Ethereum and smart contracts

Ethereum [26] is a blockchain-based cryptocurrency. Similar to other such currencies it relies on its decentralized structure to reach a consensus on a public ledger containing blocks of transactions.
The feature that makes Ethereum suitable for implementing IKP are smart contracts. While originally defined as a digital version of contracts, Ethereum takes them one step further and allows executing arbitrary code in contracts which means the consensus also has to enforce correct contract execution. It achieves this by introducing a new type of account, the contracts, which are created by sending a transaction from an externally controlled account to the empty account with the code to be executed as data. While smart contracts can contain programs and state they can only be executed or react to something if a transaction is sent to the contract account, thus making code execution in the system dependent on actions of externally controlled accounts. Computation power, which is essentially a state change of the contract, is paid for in a seperate currency from Ethereums currency, called gas. The price for gas is relatively low, however each contract has to have a self imposed gas limit.
However smart contracts are not fool proof, can have bugs and can be attacked [3]. An example of this is the DAO hack [17], where a large contract had severe problems that caused unwanted money transfers. In the end this lead to a fork of the blockchain, which is generally speaking undesirable because it disrupts the value of the currency and reduces trust in the consensus mechanism.

3. INSTANT KARMA PKI

This section is going to describe the Instant Karma PKI [15] system with all its components and afterwards discuss and evaluate it.

3.1 Analysis

To describe a system such as IKP it is important to keep in mind what problems exist and how it attempts to approach them, the authors of IKP identified the following problems [15]. Certificate Transparency, as briefly described in Section 1, doesn't solve important problems that exist in todays PKI infrastructure.
One flaw of CT specifically is that the list of authorized logs is centralized, thus adding another point of failure, the list of trusted logs shipped with browsers. Other log-based enhancements have tried to solve this [12], however none have been implemented, so actual security of alternatives is hard to verify.
Another obvious problem is running logs and monitors. Running logs is expensive for actors not participating in the PKI, specifically anyone but CAs, and some logs in the current CT system are run by CAs who were compelled by Google to do so due to security incidents [20]. Furthermore running monitors is best done by domains themselves since they know best what authorizes a certificate for their domain, while this requires additional setup, services like SSLMate's Cert Spotter, which is available on github [22], make this easier even for smaller actors.
An important issue is the difficulty of actually reporting unauthorized certificates. Only the CAs themselves can currently revoke certificates within a reasonable timeframe and reporting to them means manual effort for the detector. Other possibilities for the detector would be contacting browser vendors and getting the root certificate revoked, however rolling out browser updates takes testing and time and thus won't be finished in a short amount of time either. An example for where this works is the repeated misbehavior of Symantec, who Google decided to take action against by severely limiting Symantecs root certificates in Chrome [21].
Another observable issue is that CAs need to invest more in their security and their verification process. As seen in Section 1 a lot of CAs have been the target of hackers who then proceeded to execute Man-in-the-middle (MitM) attacks, thus increasing their security is the obvious move. Some CAs also seem to take the verification of ownership very lightly and gave intermediate certificates to companies that don't enforce any verification [13]. They need to be given incentives to invest in their security, and the verification of domains and of companies they sell certificates to.
With these problems in mind, IKP's goals are to add financial incentives for CAs to behave correctly, besides the monetization of certificates, to automate verification of suspicious certificates and to reward the detectors accordingly and to achieve this in a decentralized way [15].

3.2 Overview

This section defines the main goals as identified in the IKP paper [15]. The main idea of IKP is to automate a decentralized system that creates financial incentives for handling CA misbehavior. To do so it is important to define CA misbehavior, so it has to introduce a concept that allows the definition of authorized certificates. To also add automated financial incentives for reporting certificates, IKP needs the ability to evaluate those certificates and it has to provide a way to respond in case the evaluation results in the identification of CA misbehavior.
These ideas result in a system that should fulfill the following properties:

- Auditability of the information in the IKP system that define authorized certificates.
- Automation of reactions to CA misbehavior without any third parties
- Financial incentives that ensure actors who show good behavior get rewarded
- Punishment of CAs for misbehavior resulting in discouragement of more misbehavior

Before going into details, actors in this system are established and possible adversarial behavior of those actors is described. The three obvious actors, are the ones that interact in the TLS protocol's system, namely domains, CAs and clients. Another important actor is the detector, who reports suspicious certificates. Since IKP is based on the idea of adding financial incentives to the PKI, the goal of all actors is a positive Return of Investment (ROI). For CAs

generally speaking this means issuing an unauthorized certificate but not being penalized for it or receiving more payments than penalties. Domains can act malicious via collusion attacks together with detectors and CAs, and detectors can try to report every certificate they come across and report it, hoping for one of them to actually be unauthorized and thus receiving rewards.
In TLS the domains buy certifications of their public keys in order to verify their identity to clients during the TLS handshake and establish a secure connection for transferring data. In order to achieve IKP without impact on the existing infrastructure, a new entity is introduced: the IKP authority. As seen in Figure 2 it enables domains to register so called Domain Certificate Policies (DCP), which are a concept for describing CA misbehavior. DCPs allow the domain to define what makes a certificates authorized in a way that can be checked by the IKP authority automatically. CAs are enabled to register Reaction Policies (RP) to the IKP authority. These policies are bought by domains from CAs and are supposed to be the financial incentive for CAs to not issue unauthorized certificates, because if no unauthorized certificates are issued against the domain until the expiry date, the CA is rewarded. They act as an insurance for the domain against misissuance, as the domain and the detector receive rewards from the RP if an unauthorized certificate for it is detected. The IKP authority also executes the important inspection of the certificates that detectors report using the DCP that the domain published and issues payouts based on the RPs. Furthermore the authority controls a global fund to send and receive payments, as a result it has the role of a trustee to all other parties.
To register in this system all entities have to first register in the Ethereum blockchain. This is necessary in order to receive payments and also to interact with the IKP authority, since it is implemented as a smart contract and thus can only be interacted with by sending transactions to it. This is what results in a fully automated system and enables IKP to run with little supervision. Another result of the authority being a contract is that the essential part of DCPs and RPs have to be implemented as a contract too, such that the IKP authority can automatically execute them in case it has to check a reported certificate or send payments from the reaction policy.
To summarize, the IKP authority has to store DCPs and RPs in order to automate misbehavior checking, the CAs issue RPs to domains, the domains publish DCPs to articulate what authorizes a certificate for them and the detectors report suspicious certificates, thus starting the checker functionality of the IKP. The authority contract has to escrow funds and send out payments according to the results of checks of certificates.

3.3 IKP authority

This section further describes the functionality offered by the IKP authority as it is described in the IKP paper [15]. Similar to having their root certificate in web browsers, **CAs** need a way to **register** in IKP if they want to sell insurance policies. To start this they have to register an externally controlled account in Ethereum and then send a transaction to the IKP contract with the fields depicted in Figure 3. The *CA name* identifies it, the *Valid From* field specifies at what point the registration information is valid, the *payment account* is the address of the CAs Ethereum account,

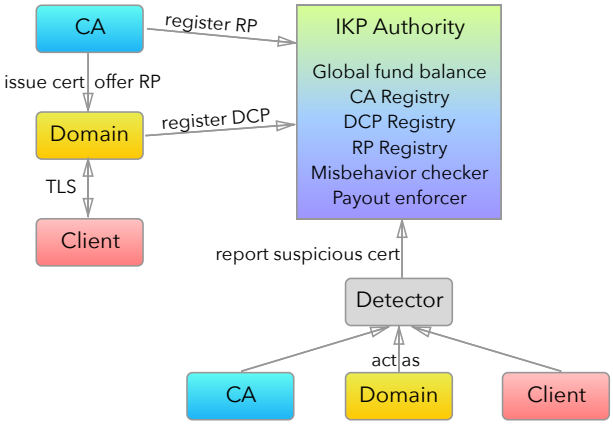

Figure 2: Overview of IKP showing its entities and corresponding functions. Adapted from [15].

note that it does not necessarily have to be the account the registration transaction came from. The *public keys* are the CAs signing keys that are contained in their root certificate, the *update keys* are an optional security measure that allows CAs to define other keys in case their primary account is compromised and lastly the *update threshold* defines how many update key signatures are needed to update the registration information. It is important to see that while the update mechanic adds a certain amount of security, it is not a complete protection against compromise, private keys still have to be kept secret. However since CAs already have to store their certificate signing keys in a secure way, securely storing update keys and the Ethereum key should not be a problem.

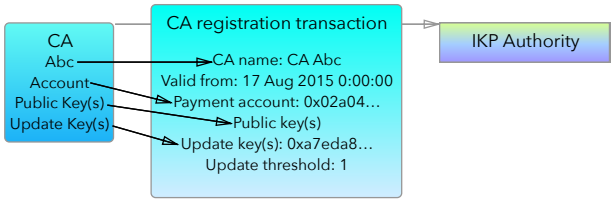

Figure 3: Overview of CA registration in IKP. Adapted from [15].

Domains have to be able to **register** in the IKP system too, since they have to issue the DCPs which are crucial to the whole system. The idea is to use DCPs not only to describe authorized certificates but also to act as a way of registering. The domain has to register similar information to CAs, this is described in detail in Section 3.4.
The IKP also offers a fair exchange mechanism for payment transfers during **RP issuance**. The procedure is as follows: First the domain sends the payment for the price/fee for the RP and a hash over the RP to the IKP authority, then the CA creates and sends the RP to the authority. The authority then verifies that the sent RP corresponds to what the domain is expecting to buy from the CA and that certain payout constraints are fulfilled. If anything goes wrong all payments are returned and the process is cancelled. If everything is correct the IKP authority will send the price the domain paid for the RP to the CA and all funds in the

doi: 10.2313/NET-2017-09-1_05

transaction that created the RP are transferred to a global fund that the IKP authority maintains.
With this, after the domain and the CA negotiated the contents of the RP, the IKP acts as an escrow service for the associated payments. This escrow functionality can also be used to issue certificates, since both parties who participate in certificate issuance, the CA and the domain, already manage Ethereum accounts to participate in the IKP system, it makes sense to offer this service as the IKP authority itself is an easily auditable smart contract.
Furthermore the IKP has to accept **reports of misbehavior** from detectors. This usually happens in the form of sending a suspicious certificate and Ethereum account information. The detector has to pay a fee for reporting certificates in order to discourage sending every single certificate a detector encounters and hoping one of them is unauthorized.
Another important part of this is a so called pre-report containing the reporting fee and a hash of the certificate and a secret. This is enforced so that blockchain miners cannot replace the detectors name with their own and thus receive the rewards for detecting a malicious certificate. This is called commitment hash and the commitment is opened after a certain number of blocks are mined by sending the actual certificate and the secret. Note that the reward for a detector is supposed to be negotiated between domain and CA when buying a RP and furthermore has to be larger than the initial reporting fee. The reward is only sent out if the IKP authority detected misbehavior.
If misbehavior is detected, not only is the detector paid, the reaction program also contains **payouts** to the domain and the CA. The financial transactions are executed by the IKP authority. The penalty in the CA occurs in the form of less payout than it would have received without misissuance.

3.4 Domain Certificate Policies

This section describes the design goals for DCPs, the actual implementation and functionalities that the checker program has to offer, as described in the IKP paper [15]. The last paragraphs give information about the functionalities the IKP authority has to provide and leverage in order for DCPs to function.
Domain Certificate Policies are an important aspect of the IKP architecture since they allow domains to publish what properties an authorized certificate has to fulfill, which then defines CA misbehavior. Following are the design principles the IKP creators had in mind when developing DCPs.
Firstly the information contained in DCPs that allows checking a certificate and results in the assessment of the certificate, is only determinable by domains themselves. Thus IKP allows only domains themselves to specify what authorizes a certificate for them. As a result it is possible to move away from terminology established in CT, the concept of a suspicious certificate, and concretely define authorization criteria. This also enables detectors who know these criteria to report unauthorized certificates with corroboration from domains themselves.
Secondly to enable detectors to determine criteria for authorized criteria, DCPs are stored in a decentralized way. This is enabled through the blockchain and enables the information to be globally consistent through the consensus mechanism. This also allows the authenticity to be affirmed by anyone, thus making determining authorization of certificates globally consistent.
Lastly the criteria to determine authorization have to be as expressive and detailed as possible. This allows for varied policies and this is important since larger organizations probably have more complex requirements for their certificates. This is realized by implementing the checker program as smart contracts that returns a boolean value describing authorization, `true` if a certificate is authorized, `false` otherwise. An interesting observation is that while programs can be arbitrary since Ethereum offers turing-complete languages to program contracts, the gas cost for certain operations is relatively high and reaches the upper limit of compute power contracts are allowed to use. An example for this would be parsing X.509 certificates [7], the type of certificate used in TLS. This resulted in the current implementation of IKP having to employ different parsing techniques. However since the method provided by the checker contracts does not change the program's state, they only have an initial gas cost for deploying them and afterwards are free to use by the IKP authority to check whether a specific certificate is authorized or not.

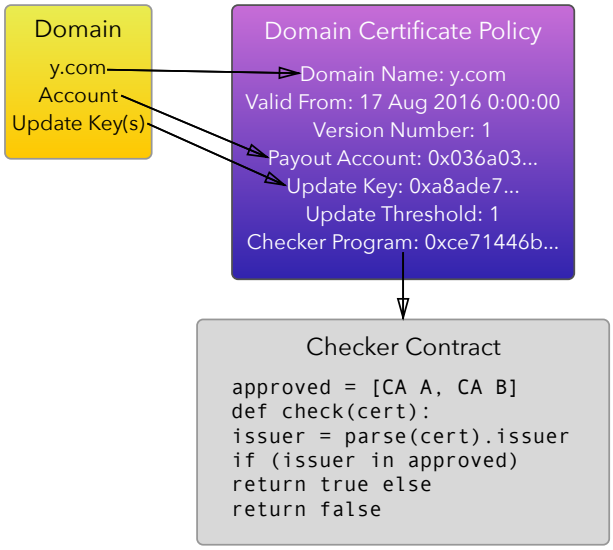

Figure 4: Overview of a Domain Certificate Policy. Adapted from [15].

DCPs themselves, or rather the transaction that the domain has to send, have to contain information about the policy itself, such as *Valid from* and *Version number* fields, but also information about the domain, since DCPs are to also function as a domain registration as described earlier. All fields that have to be part of the DCP can be seen in Figure 4. Information for domain identification are the domain name, payment account and update key fields. The *Domain name* is the domain's DNS name, the *Payout account* is the domain's Ethereum account it intends to send and receive payments in the IKP system from and the *Update keys* allow the domain to recover from a compromised or lost payment account. The *Update threshold* defines how many signatures of update keys are needed to update information in that case and lastly the *Checker program* binds a specific checker contract to the DCP. This checker contract has to provide a method `check()` that the IKP authority can call when checking a specific certificate. Identifying the policy is

doi: 10.2313/NET-2017-09-1_05

implemented via the *Valid from* field, which specifies when the DCPs comes into effect, the *Version number* field which identifies the new DCP, but only if the checker program is replaced or modified, and the checker program, which contains the account key of the checker program contract. The *Valid from* and *Version number* fields are further used to bind a specific checker program to a RP, since running every single DCP the domain published, in order to verify a reported certificate, is time intensive and could lead to conflicting information. The specific interactions are described in Section 3.5.
To realize the functionality of DCPs the IKP authority has to be able to offer a way to prove ownership of a domains during registration. To achieve this either the existing DNS or TLS infrastructures are leveraged. The domain signs its name and DCP either with its DNSSEC private key and includes a signature chain to the DNS root or in the case of TLS it signs with its TLS private key and includes a chain to a root CA. The first option is the preferred way of handling this since the DNS root has less of a history of security incidents. However, according to measurements by the IKP authors [15], DNSSEC is not widely adopted. Thus the TLS approach is the more realistic one, however it requires the IKP to have a list of accepted root CAs just like browsers. Thus it is important to keep the amount of trusted CAs minimal, so that DCPs cannot be registered with rogue certificates. For updates the IKP authority has to verify signatures signed with the domain's update keys and ensure the minimum amount specified by the domain is met.

3.4.1 *Examples for DCP checker contract functionality*

This section will describe possible functionality of checker contracts and operational functionality the IKP authority has to offer to run the system. As defined earlier, the policies, implemented as checker programs, should be expressive. The following example implementations can be combined with each other to achieve a set of desired criteria for a domain's certificates. Since contracts can call each other's check method, if a set of criteria is already implemented by another domain they can be included in the domain's own contract using boolean relations such as AND or OR.
The standard for certificates in TLS is X.509, so the fields available in such a certificate are provided as parameters through parsing. The first obvious kind of program are whitelists for CAs. The contract takes the *Issuer Name* field and matches it against a list of authorized CAs. Another whitelist can be implemented by checking the key identifier contained in the certificate. Similar to public key pinning, where domains specify a list of trustable keys, the *Subject Public Key Info* field can be extracted and matched against a list of authorized and pinned keys. This is a different approach compared to current implementations of certificate pinning [11] because of the decentralized storage of the list. Short-lived and wildcard certificate rules are also supported by extracting the *Not Before* and *Not After* fields to determine the lifetime of a certificate and extracting *Subject Name* to make sure * does not appear. Wildcard certificates specifically enable MitM attacks on all subdomains, since wildcard certificates are valid for every single subdomain. As in CT's implementation it is possible to enforce that a certificate has been logged, similar to the signed certificate timestamp (SCT).

3.5 Reaction Policies

This section explains the goals for designing RPs, their implementation and lastly gives an overview over the different scenarios that involve CA misbehavior or good behavior and how the financial rewards work in these situations, as described in the IKP paper [15].
Unlike Domain Certificate Policies, Reaction Policies only serve a single purpose, which is acting as an insurance for the domains against misbehavior. However just like with DCPs there are three main design goals. They are intended to prevent unwanted incentives and consequences.
Firstly RPs should be independent from certificates, because while they are both sold by CAs to domains, they are supposed to be an addition to certificates and are supposed to protect domains against misbehaving CAs, so binding them to one specific certificate doesn't make sense.
Secondly RPs are issued for a specific DCP, specifically they are bound to a single version of a DCP, which means there is exactly one checker program the IKP authority has to check against if a certificate issued by the CA that issued the RP is reported. This means the domain has to be registered via a DCP in the IKP system before being able to purchase RPs.
Thirdly a RP is only valid against a single occurrence of CA misbehavior. This is mainly done because the IKP authority has to make sure there are enough funds available to issue payouts. A domain can however hold multiple reaction policies, so the only limiting factor is the transaction that has to take place in order to register the RP in the IKP authority.

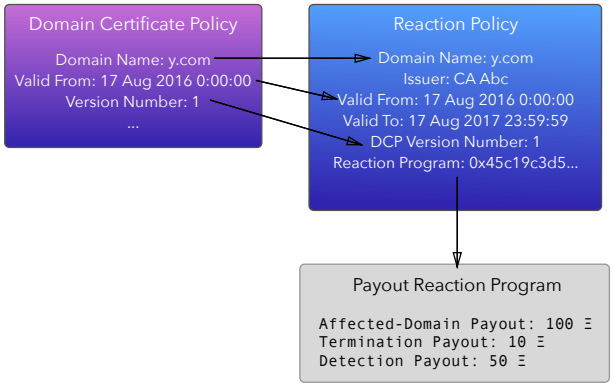

Figure 5: Overview of a Reaction Policy. Adapted from [15].

As with DCPs, RPs have to be sent as a transaction to the IKP authority. Figure 5 shows the different fields the policy has to include. The *Domain name* field, which identifies who the RP was issued to, the *Valid from* field, which describes the time from which the RP becomes active and the *DCP version number*, which binds the triggering of an RP to a specific checker contract, all serve the same purpose as in DCPs and bind a reaction policy to specific domain policy. The *Issuer* field serves as an identification of the CA that issued the policy, the *Valid to* field which denotes when the RP runs out without any occurrence of misbehavior and lastly the *Reaction program* contains the address to a reaction contract account. The reaction program, similar to the checker program, has to define specific methods. The `trig-`

`ger()` method is called when an unauthorized certificate is reported via the IKP system for the domain specified in the policy. If that is the case, the IKP authority also records the date when misbehavior occurred. The `terminate()` method is called by a domain in case a CA issued an unauthorized certificate and the domain wants to terminate the RP early, because it lost trust in the CA. In this scenario the IKP authority has to compare the *Valid from* field with the date of misbehavior occurrence. The `expire()` method is what ends the contract at the date specified in the *Valid to* field and is called by the CA that issued the RP.
As seen in Figure 2 the IKP authority acts as a registry for RPs. This is implemented as a mapping of domains and a list of their active RPs. This list is ordered by the *Valid to* field. This ensures that the RP that expires the soonest is triggered first.

3.5.1 Payouts and constraints

The reaction program in Figure 5 is just a sample of what can happen as a reaction. While the program has to provide three methods, it is not defined what they do, which differentiates it from checker programs that have to return a boolean value. However the overall goal of IKP, as defined in Section 3.2, is creating financial incentives, so the only contract discussed here and the original paper [15], are payout reaction programs. The goal was to keep this as generic as possible and thus specific payment amounts are not enforced or given.
The **affected-domain payout** is the amount the domain receives if any CA that is registered in IKP issues an unauthorized certificate for the domain. This payment does naturally not occur if the CA is not registered with the IKP authority. It is supposed to compensate the domain for the risk of MitM attacks.
The **termination payout** is a payment that gets split between the CA that issued the RP and the domain that bought it. This payment occurs whenever the reaction contract is terminated early, there can be multiple reasons for this, which will be explained in Section 3.5.2. The payment is split proportionally to the amount of time left in the RP's validity period in such a way that the domain receives less money the longer the RP is active. The amount the domain receives is bound by a minimum that is defined globally in IKP and the total amount defined in the payout contract.
The **detection payout** is the financial incentive for the detectors for monitoring logs and CA operations and for reporting unauthorized certificates they may find. Note that this payment is only made if the reported CA is registered in the IKP system and as such can be higher than the initial fee a detector has to pay for reporting since the amount can be imposed on the misbehaving CA. This also means if the CA is not registered in IKP the detector simply gets the initial fee paid back.
To issue an RP the domain and the CA negotiate the terms of it outside the IKP system. If the RP is negotiated as a payout contract, they negotiate the price/fee and just described payouts. There are two constraints that must be held up: The affected-domain payout plus the minimum termination payout for the domain must be larger than the price. This is a measure against collusion attacks between domains and CAs or domains and detectors and guarantees a negative ROI if all payouts are summed up. The price must be larger than the termination payout itself, so that an issuing CA can still profit from a RP that is terminated, when the issuing CA wasn't the one that misbehaved. As mentioned in Section 3.3 the detection payout must be larger than the fee the detector has to pay when reporting certificates.

3.5.2 Scenarios of (mis)behavior

This next section lays out possible scenarios that describe reactions to CA misbehavior and correct behavior and how aforementioned payouts are involved, as described in the IKP paper [15].
If a CA issued an RP for a domain and there are no occurrences or detections of misbehavior by this CA during the validity time, the RP simply expires and the issuing CA receives the money it escrowed by the IKP authority for the CA. This escrowed funds are not defined in the final version of the IKP paper [15], however in the first version it is described as a fraction, between zero and one, of the sum over affected-domain, termination, detection and global fund payouts.
If a domain terminates an RP early, for example if the issuing CA misbehaved against someone else, it is paid its fraction of the termination payout and the CA is paid the escrowed funds minus the domains termination payout.
For the next part the concept of internal and external misbehavior is introduced. Since the IKP system can only penalize CAs that are registered with it, it distinguishes between internal, where the offending CA is registered with the IKP authority and external misbehavior, where the CA is not part of IKP.
In the case of internal misbehavior we distinguish between the CA that misbehaved and the CA that issued the RP since the RPs are ordered by validity ending time, and thus the case, where the misbehaving and the issuing CA are the same entity, is rare. The detector sends the reporting fee to the IKP authority, which then detects the misbehavior. It then makes the misbehaving CA pay the affected-domain payout and the detection payout to its global fund. Afterwards it pays the domain the affected-domain payout and its part of the termination payout, which will be relatively close to the minimum payout that is globally defined, since the oldest contracts are triggered first. The detector receives the detection payout and the CA that issued the RP receives all escrowed funds minus the domain's fraction of the termination payout.
For external misbehavior penalization of the CA that misbehaved is impossible. Thus the detector only gets its reporting fee back and no payout. The domain again receives a fraction of the termination payout and the RP issuing CA receives the escrowed funds minus the domain's split.
Lastly if an RP expires, which means the time specified in the *Valid to* field is in the past and no misbehavior happened, the IKP authority removes the RP from the mapping mentioned earlier and sends any payments made by the CA during issuance, which were escrowed, back to it.

3.6 Evaluation

After describing IKP, this paper now evaluates IKP and describes possible problems and implementation challenges. It is important to note that while there seems to be a github for IKP, `https://github.com/syclops/ikp`, it is apparently not available to the public.

doi: 10.2313/NET-2017-09-1_05

3.6.1 Weaknesses of the architecture

Firstly the negotiation and effectiveness of RPs heavily depend on the size of the participating entities [15, p. 7]. One strategy that CAs specifically could employ is minimizing the detection payouts. While this wouldn't work when negotiating with big organizations wanting to participate in IKP, it certainly works with smaller actors. This could cause all RPs a domain owns to have low detection rewards and thus can remove financial incentives from reporting rogue certificates for all small domains.
Secondly CAs have to willingly participate in the IKP system in order for it to be effective. This means the financial incentives have to outweigh the risk the CAs incur by being openly penalized for misbehavior.
The system allows the fair exchange mechanism used for RP issuance to be used for certificate issuance too, however this adds more bloat to the system and does not seem to make sense financially since cryptocurrencies are rather unstable.
Lastly if for example a single person detects an unauthorized certificate, there's an initial hurdle for them by having to register in Ethereum in order to send transactions to the IKP authority. While this system is clearly enhancing financial incentives for log operators and monitors, it does not put a focus on single detections.

3.6.2 Problems and risks incurred by the implementation in Ethereum

Another risk for CAs are vulnerable contracts. As seen in the DAO hack [17], a contract cannot be updated easily if it is found to be vulnerable and thus a CA could potentially lose escrowed money. Another example for this is the Parity multisig bug, which recently occurred due to method passthrough functionality in Ethereum contracts [5].
The current version of IKP is also not implementable due to RSA signature verification [15, p. 12], which is presumably needed for update keys, not being a part of the current Ethereum implementation for smart contracts. It heavily reduces the cost from 3000 gas to 200. There exists a proposal on the Ethereum proposal github [4], however it has been open for over a year as of the writing of this seminar paper and noone is assigned to it.
Another problem that is based on Ethereum is that while smart contracts are written in a turing complete language and thus can produce arbitrary programs, there is a maximum limit on gas, which limits actors with complex checker contracts. Furthermore it can take a significant amount of effort to formulate a contract for large entities with complex requirements for certificate authorization.
An aspect that is important to consider is privacy. In a system such as IKP that enables an insurance system, transactional privacy is a desirable property. Ethereum lacks this feature, however the next major release of Ethereum, Metropolis, is supposed to include privacy enhancing features [6].
The recent development of a decompilation tool called Porosity [25] has shown that reverse engineering smart contracts is a focus in research. This can become a serious problem for the security of checker contracts, since they contain sensible information. This could for example allow an attacker to decompile a domain's checker contract and attack specific CAs listed in the contract, if it is implemented as a simple whitelist.

4. CONCLUSION

Instant Karma PKI is meant to add financial incentives to the PKI that disincentivize misbehavior and reward the absence of it. A blockchain is used to add natural financial incentives and keep the access to the whole system decentralized. The choice of Ethereum specifically allows for a large part of the system to be completely automated. In order to define misbehavior, it allows the only actor that truly knows when a certificate for itself is malicious, the domain, to define what authorizes certificates issued for it. While lacking in implementation, the system is well thought out and is relatively cheap to run due to low gas costs in Ethereum. This system does not solve the problem of misbehavior occurring itself, however doing so would probably require a full rework of the whole PKI and reworking systems on the internet is difficult and adoption for new standards is low, this can be seen with IPv6 for example. It also cannot function properly if no CA willingly joins the system, however the hope is that the financial incentives outweigh the risk of being penalized for misbehavior.

5. REFERENCES

[1] Certificate Transparency. https://www.certificate-transparency.org. Accessed: 2017-07-01.

[2] H. Adkins. An update on attempted man-in-the-middle attacks. https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html, August 2011. Accessed: 2017-07-01.

[3] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on Ethereum smart contracts. Cryptology ePrint Archive, Report 2016/1007, 2016. http://eprint.iacr.org/2016/1007.

[4] A. Beregszaszi. Support RSA signature verification. https://github.com/ethereum/EIPs/issues/74, March 2016. Accessed: 2017-07-01.

[5] L. Breidenbach, P. Daian, A. Juels, and E. G. Sirer. An In-Depth Look at the Parity Multisig Bug. http://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/. Accessed: 2017-07-26.

[6] V. Buterin. Ethereum Research Update. https://blog.ethereum.org/2016/12/04/ethereum-research-update/. Accessed: 2017-07-27.

[7] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008. Updated by RFC 6818.

[8] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919.

[9] B. Dowling, F. Günther, U. Herath, and D. Stebila. Secure Logging Schemes and Certificate Transparency. Cryptology ePrint Archive, Report 2016/452, 2016. http://eprint.iacr.org/2016/452.

[10] P. Ducklin. The TURKTRUST SSL certificate fiasco - what really happened, and what happens next? https://nakedsecurity.sophos.com/2013/01/08/the-turktrust-ssl-certificate-fiasco-what-

happened-and-what-happens-next/, January 2013. Accessed: 2017-07-01.

[11] C. Evans, C. Palmer, and R. Sleevi. Public Key Pinning Extension for HTTP. RFC 7469 (Proposed Standard), Apr. 2015.

[12] T. H.-J. Kim, L.-S. Huang, A. Perrig, C. Jackson, and V. Gligor. Accountable Key Infrastructure (AKI): A Proposal for a Public-key Validation Infrastructure. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 679–690, New York, NY, USA, 2013. ACM.

[13] A. Langley. Maintaining digital certificate security. https://security.googleblog.com/2015/03/maintaining-digital-certificate-security.html, March 2015.

[14] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962 (Experimental), June 2013.

[15] S. Matsumoto and R. M. Reischuk. IKP: Turning a PKI Around with Decentralized Automated Incentives. In *S&P'17: 38th IEEE Symposium on Security and Privacy (Oakland)*, 2017.

[16] J. Menn. Key Internet operator VeriSign hit by hackers. http://www.reuters.com/article/us-hacking-verisign-idUSTRE8110Z820120202, February 2012. Accessed: 2017-07-01.

[17] R. Price. Digital currency Ethereum is cratering because of a $50 million hack. http://www.businessinsider.de/dao-hacked-ethereum-crashing-in-value-tens-of-millions-allegedly-stolen-2016-6, June 2016. Accessed: 2017-07-01.

[18] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000. Updated by RFCs 5785, 7230.

[19] P. Roberts. Phony SSL Certificates issued for Google, Yahoo, Skype, Others. https://threatpost.com/phony-ssl-certificates-issued-google-yahoo-skype-others-032311/75061/, March 2011. Accessed: 2017-07-01.

[20] R. Sleevi. Sustaining Digital Certificate Security. https://security.googleblog.com/2015/10/sustaining-digital-certificate-security.html, October 2015. Accessed: 2017-07-01.

[21] R. Sleevi. Intent to Deprecate and Remove: Trust in existing Symantec-issued Certificates. https://groups.google.com/a/chromium.org/d/topic/blink-dev/eUAKwjihhBs/discussion, March 2017. Accessed: 2017-07-01.

[22] SSLMate. Cert Spotter. https://github.com/SSLMate/certspotter. Accessed: 2017-07-26.

[23] T. Sterling. Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard. https://technet.microsoft.com/library/security/ms01-017, March 2001. Accessed: 2017-07-01.

[24] T. Sterling. Second firm warns of concern after Dutch hack. https://www.yahoo.com/news/second-firm-warns-concern-dutch-hack-215940770.html, September 2011. Accessed: 2017-07-01.

[25] M. Suiche. Porosity: A Decompiler For Blockchain-Based Smart Contracts Bytecode. DEF CON 25, July 2017.

[26] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. Technical report, 2015.