Chapter 11
Cryptographic Key Distribution and Management

Martin Rublík
University of Economics, Slovakia

ABSTRACT

Cryptographic key distribution and management is one of the most important steps in the process of securing data by utilizing encryption. Problems related to cryptographic key distribution and management are hard to solve and easy to exploit, and therefore, they are appealing to the attacker. The purpose of this chapter is to introduce the topics of cryptographic key distribution and management, especially with regards to asymmetric keys. The chapter describes how these topics are handled today, what the real-world problems related to cryptographic key distribution and management are, and presents existing solutions as well as future directions in their solving. The authors present the cryptographic key management and distribution problems from a multidisciplinary point of view by looking at its economic, psychological, usability, and technological aspects.

INTRODUCTION

Encryption is especially useful for providing security related services such as protecting data integrity and confidentiality, or supporting authentication processes (ITU-T, 1991). Though encryption is a vital part of a security system, it should be viewed only as one of the countermeasures. Encryption protection can be circumvented or broken at many levels. Encryption systems can be broken at design or implementation level, or can be simply bypassed. There exist several examples and proof-of-concepts of applied cryptography related attacks at design and implementation level, but most of the real-world attacks are simply bypassing the cryptography protection. Cryptographic protecion is circumvented either by attacking at application layer (Langley, 2013) where the data is unencrypted, or by attacking the key management and distribution infrastructure (Leavitt, 2011).

In this chapter we present why it is appealing for the attackers to take advantage of weak cryptographic key management and distribution. Hence we focus on description of existing approaches to symmetric and public key distribution and management from the user's point of view (usability) and from the security infrastructure provider's point of view (economics).

In the first part of the chapter we present basic topics related to cryptographic key distribution and management. We outline key management life-cycle and describe the dependencies between its phases and security services as defined by (ITU-T, 1991). We also explain important factors

DOI: 10.4018/978-1-4666-5808-0.ch011

that affect the approach to key distribution and management.

In the second part we briefly describe different key distribution and management approaches relevant to symmetric cryptography. We mention especially pre-shared keys and key distribution centers.

Because public key cryptography is commonly used for key distribution of symmetric keys, we focus extensively on public key distribution and management in the third part of the chapter. In particular, we describe centralized, decentralized and hierarchical approaches to public key distribution and management.

In the fourth part we compare particular approaches to public key distribution and management and outline their core characteristics, advantages and issues.

Finally, in the last part of the chapter, we further examine existing and future solutions to the issues of X.509 public key infrastructure. We chose X.509 public key infrastructure because it is most widely deployed Internet cryptographic key infrastructure and provides key management services for common network encryption protocols such as SSL/TLS, IPSec and S/MIME.

GENERAL CONCEPTS IN KEY MANAGEMENT AND DISTRIBUTION

In order to protect data integrity and confidentiality by using encryption the involved parties must share cryptographic keys. Modern encryption systems are designed in a way that protects the data if the encryption algorithm is known and encrypted data is available to the attacker. Many of the systems are designed to further protect the data even if the attacker knows the encryption algorithm, knows the encrypted data, and can decrypt some of the data. It is however impossible to protect the data if the attacker can gain access to the entire set of cryptographic keys. Hence, key management and distribution is crucial for cryptographic protection.

According to (NIST, 2007) cryptographic key management lifecycle consists of several phases. Their simplified illustration is expressed by Figure 1.

Especially hard part of key management lifecycle is trust establishment and key distribution in pre-operational phase. In these processes involved parties need to prove the possession of cryptographic keys, and exchange them in a secure manner.

There are several approaches applicable to distribution of cryptographic keys and trust establishment. Their suitability depends on a number of factors. Factors that are especially important include: type and purpose of performed cryptographic transformation and security service that requires this transformation, overall number of entities that are part of the process, the skills and psychological characteristics of the users and system administrators involved in the process. These factors influence the trade-off between usability, security, and costs of approach to key distribution and management.

Security services as defined by (ITU-T, 1991) include: authentication, data confidentiality, data integrity and non-repudiation. These services affect key management and distribution process in its several phases, especially in: key generation phase, trust establishment phase, revocation phase, escrow and backup phases. For example consider that non-repudiation service should be provided. The cryptographic keys supporting non-repudiation service need to be generated by its owner, or at least the keys need to be generated in a way that provides sufficient assurance that the private key couldn't been used by anyone except its owner. Detailed illustration of dependencies is provided by Table 1.

Core aspect affecting the overall scale of cryptographic key distribution infrastructure is the number of users employing the cryptographic keys. It is evident that it is much easier to deploy and maintain small amount of cryptographic keys, than to operate a large scale Internet wide cryptographic key distribution infrastructure.

Figure 1. Key management phases

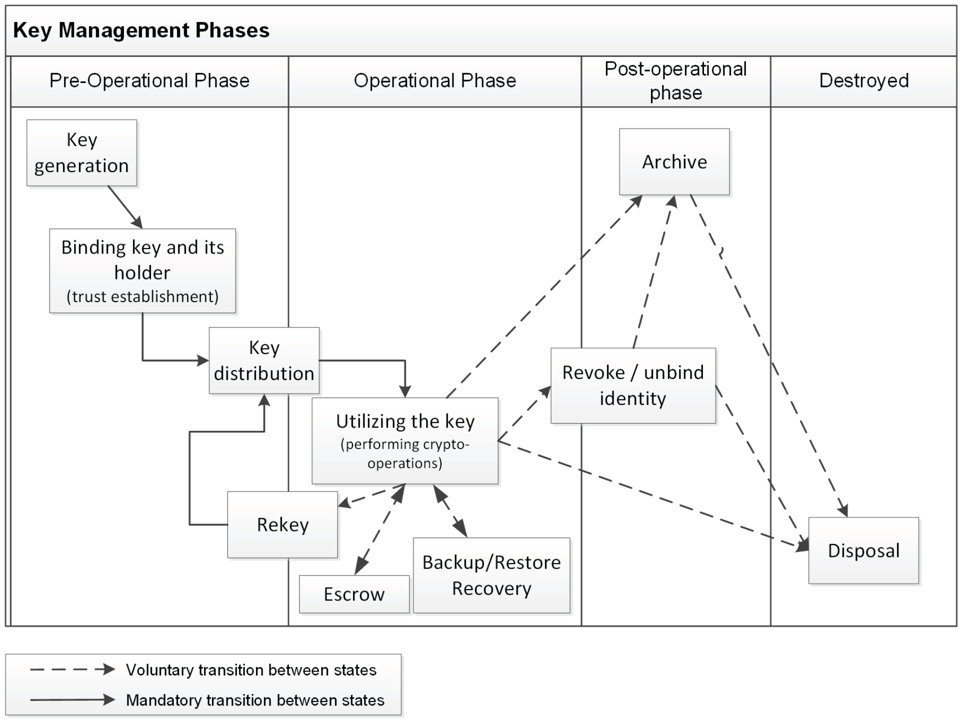


Table 1. Dependencies between key management phases and (chosen) security services

Security Service Key Management Phase Affected	Authentication	Data Confidentiality	Non-Repudiation
Key generation	Sufficient assurance that the private key can be used only by its holder should be provided.	Key generation should be done in a secure way that prevents the attacker from gaining access to private key.	Sufficient assurance that the private key can be used only by its holder must be provided.
Binding key and its holder's identity	The binding should provide a unique link to the holder's identity (with respect to information systems that make use of authentication service). Pseudonyms can be used.	Unique link to holder's identity is not required. Pseudonyms can be used.	Holder's identity and cryptographic keys should be associated in a legally binding way. Pseudonyms should not be used unless clearly marked and traceable.
Escrow / Key Recovery	Key escrow/recovery should not be used.	Key escrow/recovery might be desirable especially in enterprise environments.	Key escrow/recovery must not be used.
Backup and restore	Key backup and restore is not crucial.	Key backup and restore is important especially for protecting data-at-rest.	Key backup and restore is not crucial.
Revocation (Unbinding the holder's identity)	Revocation service or keys with short lifetime might be necessary in order to deal with key compromise (protection against impersonation attacks).	Revocation service or keys with short lifetime are necessary in order to deal with key compromise (protection against man-in-the-middle and impersonation attacks).	Timely revocation service is necessary in order to deal with key lost and compromise situations (protection against impersonation and repudiation attacks).

In several situations (as we describe later) the problem of trust establishment and key distribution is left to the users. The overall security of key distribution is therefore no more secure than the decision made by a specific user. In these cases technical skills and psychological characteristics of involved users and system administrators affect the overall security significantly. Especially psychological characteristics influence how the risk is perceived by the users that are either general users or experts in computer systems (Camp, Asgharpour, & Liu, 2007).

Cryptography and cryptographic key management is even less intuitive to general users than other topics in computer security such as computer viruses and network worms. Unfortunately, survey (Kienzle & Elder, 2003) indicates that the network worms and other types of malicious software do not need to be very innovative in order to be successful. As pointed out by (Furnell, 2005) this is a failure in application user interface design and not a failure on the user's side. The conclusion that usability is challenging in applications that utilize cryptography and perform key distribution is also encouredged by (Whitten & Tygar, 1999) and (Sheng, Broderick, & Koranda, 2006).

Usability and overall number of entities and their skills and psychological characteristics seem to be one of the core aspects that influence the complete security and feasibility of key distribution. For example it is feasible (Kahn, 1996) to use one-time pad encryption in very specific situations (small amount of high-valued data, small amount of involved, highly disciplined entities), despite its high and impractical requirements on key management. For more practical applications we discuss further flat, hierarchical, centralized, and decentralized approaches to trust establishment and key distribution in symmetric and public key cryptographic systems.

SYMMETRIC KEY DISTRIBUTION AND MANAGEMENT

In classical (symmetric) encryption systems involved parties share the same cryptographic key for encryption and decryption. Therefore, in order to apply protection one must establish a secure channel for key distribution. This secure channel needs to maintain *authenticity* and *confidentiality* of transmitted cryptographic keys.

In the real world this is achieved either by utilizing secure out-of-bound channels such as pre-shared keys, quantum key distribution, or by utilizing network security protocols and asymmetric encryption systems or trusted third parties.

Pre-Shared Keys

Pre-shared key is typically agreed upon all involved parties before they need to protect data by encryption. The pre-shared key is often distributed by employing out-of-bound channel, such as personal agreement or sharing a key through a telephone line. After the pre-shared key is distributed, it can be used directly for protecting the data, or it can be used for further key distribution.

Because pre-shared keys are usually exchanged by persons, they are often based on passwords or passphrases. However the passwords are not suitable as cryptographic keys because of their variable length and predictability. Encryption algorithms are designed to be fast, thus using a predictable password as a key would make them more susceptible to dictionary attacks.

In order to slow down the dictionary attacks keys are derived from passwords and passphrases by using a special function called *key stretching* function. The key stretching functions are designed to be slow, so the attacker cannot efficiently mount a dictionary attack.

In practice the pre-shared key distribution is popular in small scale environments. Pre-shared keys are used mostly for protecting network communications in local area networks and in networks with small number of nodes.

Pre-shared key distribution is a supported in many network security standards. Among others it is supported in Wi-Fi network protection standard Wi-Fi Protected Access (WPA) as well as in Internet Protocol Security (IPSec). Additionally, pre-shared keys are also supported in Transport Layer Security (TLS), but this part of TLS is not usually implemented in ordinary applications.

Typically pre-shared key distribution is not used in networks with larger scale because of management and scaling issues. Pre-shared keys are hard to manage, especially in cases when the pre-shared secret was compromised and should be changed. The scaling issues result from the fact that the pre-shared key should be different for every pair of communicating entities. This means that if we have *N* network nodes we need at least *N*(N-1)/2* pre-shared keys.

Key Distribution Centers

The scaling and management issues can be more or less solved by involving a trusted third party called *key distribution center* (KDC). KDC first sets up a unique pre-shared key with every entity that is part of the system. These keys are called *master keys*. Afterwards, when two entities want to exchange encrypted data, they use KDC services for secure key distribution. The key that is distributed to these entities by a KDC is called *session key*.

Two advantages follow from this approach. First, only *N* master keys are required for *N* communicating entities. This improves the scalability of the solution. Second, if a session key is compromised only data that is protected by that particular key are jeopardized. Because session keys should be used only for limited amount of time, this improves the security and manageability of the solution.

The key distribution center is usually implemented as described by (Needham & Schroeder, 1978) or (Stallings, 2011). We summarize the necessary steps as follows.

1. Alice and Bob set-up master keys with KDC. We denote the keys MK_A and $MK_{B.}$ This step needs occur only once per life of the master key, which last much longer than the session key.
2. When Alice wants to send encrypted data to Bob she asks KDC for a session key.
3. KDC sends the session key to Alice. This key is encrypted both with MK_A and MK_B
4. Alice sends the encrypted session key to Bob and starts the communication.

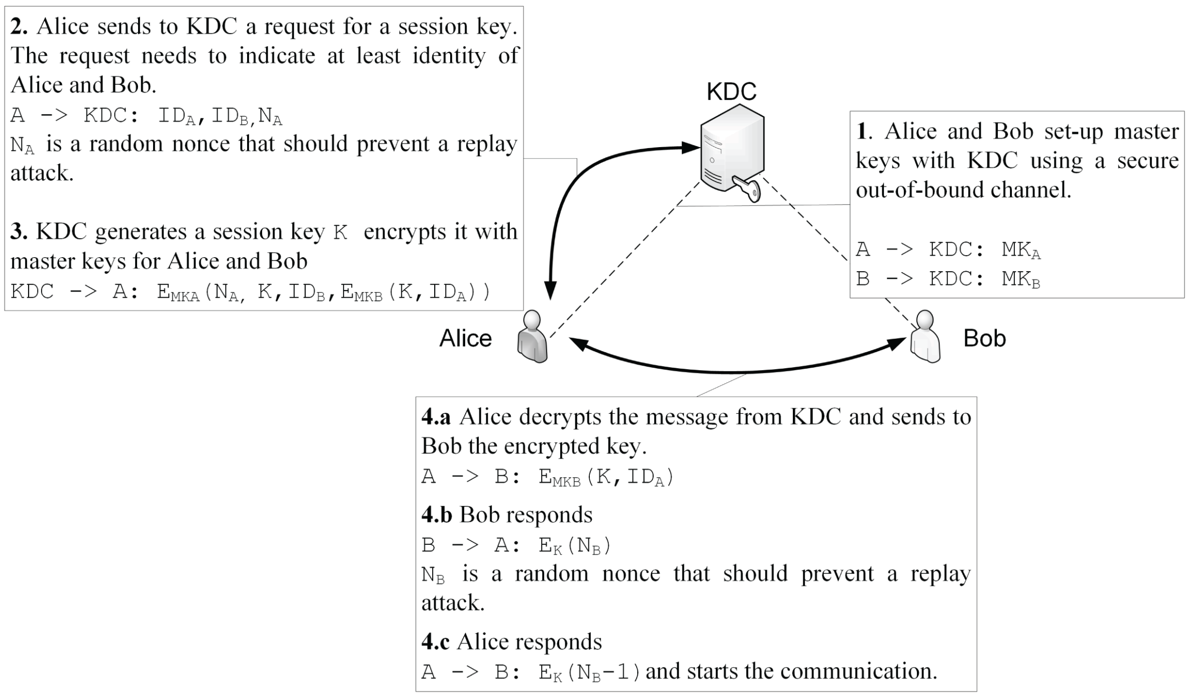
The steps involved in secure key distribution of the session key are specified in Needham–Schroeder protocol illustrated by Figure 2.

Though the Needham–Schroeder protocol illustrated by Figure 2 included replay attacks countermeasures, Denning and Sacco found a weakness in the protocol (Denning & Sacco, 1981). The protocol was fixed by introducing time stamps and time checks. With this correction the protocol serves as a basis for popular authentication and key distribution protocol Kerberos.

Kerberos is used especially for authentication in many enterprise environments because of the good support in Windows and UNIX-based operating systems. But Kerberos is supported by many popular network security protocols also for key distribution. Most notable security protocols that support symmetric key distribution by Kerberos are IPSec (Thomas, 2001) and TLS (Medvinsky & Hur, 1999).

Though Kerberos is used in large and medium networks and enterprises, it did not succeed in the Internet infrastructure. Employing a KDC for distributing cryptographic keys implies several

Figure 2. Key distribution center



2. Alice sends to KDC a request for a session key. The request needs to indicate at least identity of Alice and Bob.
`A -> KDC:` ID_A, ID_B, N_A
N_A is a random nonce that should prevent a replay attack.

3. KDC generates a session key K encrypts it with master keys for Alice and Bob
`KDC -> A:` $E_{MKA}(N_A, K, ID_B, E_{MKB}(K, ID_A))$

1. Alice and Bob set-up master keys with KDC using a secure out-of-bound channel.

`A -> KDC:` MK_A
`B -> KDC:` MK_B

4.a Alice decrypts the message from KDC and sends to Bob the encrypted key.
`A -> B:` $E_{MKB}(K, ID_A)$

4.b Bob responds
`B -> A:` $E_K(N_B)$
N_B is a random nonce that should prevent a replay attack.

4.c Alice responds
`A -> B:` $E_K(N_B-1)$ and starts the communication.

drawbacks. First, it is necessary to provide an online infrastructure for key distribution by KDC. Second, KDC is a single point of failure, both from security and from performance point of view.

These drawbacks can be partially mitigated by building a hierarchy of KDCs (Stallings, 2011). Unfortunately hierarchical KDCs still suffer from trust issues similar to issues applicable to public key distribution and management. Therefore in Internet scale the symmetric keys are usually distributed by utilizing advantages of public key cryptography.

PUBLIC KEY DISTRIBUTION AND MANAGEMENT

In asymmetric encryption systems involved parties possess a pair of cryptographic keys. One of the keys is used for encryption/verification process (*public key*) and is available to everyone. The other one (*private key*), is used for decryption/authentication process and is available only to its holder.

In order to distribute the public keys one must establish a secure channel that needs to maintain the *authenticity* of transmitted keys. Providing secure channel that maintains authenticity of transmitted keys is difficult, and means that asymmetric key distribution and management needs to take care of establishing the trust between involved parties.

The trust relationship is set up either directly between the involved parties or indirectly through a mediator. Direct trust setup is employed in naïve approaches to key distribution and management such as key continuity.

Indirect trust establishment enables more complex approach to key distribution and management. Later in this chapter we describe how indirect trust establishment makes possible centralized and decentralized key distribution.

Key Continuity

The most simplistic approach to key distribution and trust management is *key continuity*. With key continuity approach a secure channel for key distribution is provided out-of-band by verifying the public key directly by the user. This approach can be compared to pre-shared key distribution in symmetric key encryption. The entities exchange a key in a secure manner through a secure channel before encrypted communication occurs through insecure channel.

The public key is either verified directly, or indirectly by comparing its derivative created in a secure way (such as its hash). The first approach is employed[1] by popular remote access protocol SSHv1 (Ylonen, 1996). The latter is in use by secure VoIP protocol ZRTP for authenticating Diffie Hellman parameters necessary for key exchange (P. Zimmermann, 2011; Martin & Schertler, 2009). After the user confirms authenticity of the public key it is stored locally, so the user does not need to verify it in future. This approach is therefore called also Trust on First Use (*ToFU*).

Besides the mentioned world-wide deployments there exists several proof-of-concepts for incorporating key continuity in order to protect e-mail communications (Garfinkel & Miller, 2005), (Gutmann, 2004). Unfortunatelly though there is security improvement in utilizing ToFU for protecting e-mail communications (Garfinkel & Miller, 2005), even today these implementations haven't been widely deployed.

Key continuity is not deployed widely enough because it is believed that it does not scale well (Zimmermann & Callas, 2009), lacks possibility of key revocation, it is not legally binding to key holder (it does not provide non-repudiation necessary for electronic signatures), and last but not least, it is not implemented in a user friendly way, which makes it susceptible to phishing attacks (Garfinkel & Miller, 2005). The last conclusion is also supported by Peter Gutmann's paper (Gutmann, 2011). Gutmann presumes that the SSH keys are not verified by the users connecting to remote systems. His hypotheses is based on the fact that help-desk supporting personnel in two distinct large organizations were unable to recall a single case of any user ever verifying any SSH server key out-of-band. Both of these organizations were medium sized and engaged several thousand computer-literate users.

Decentralized Key Distribution

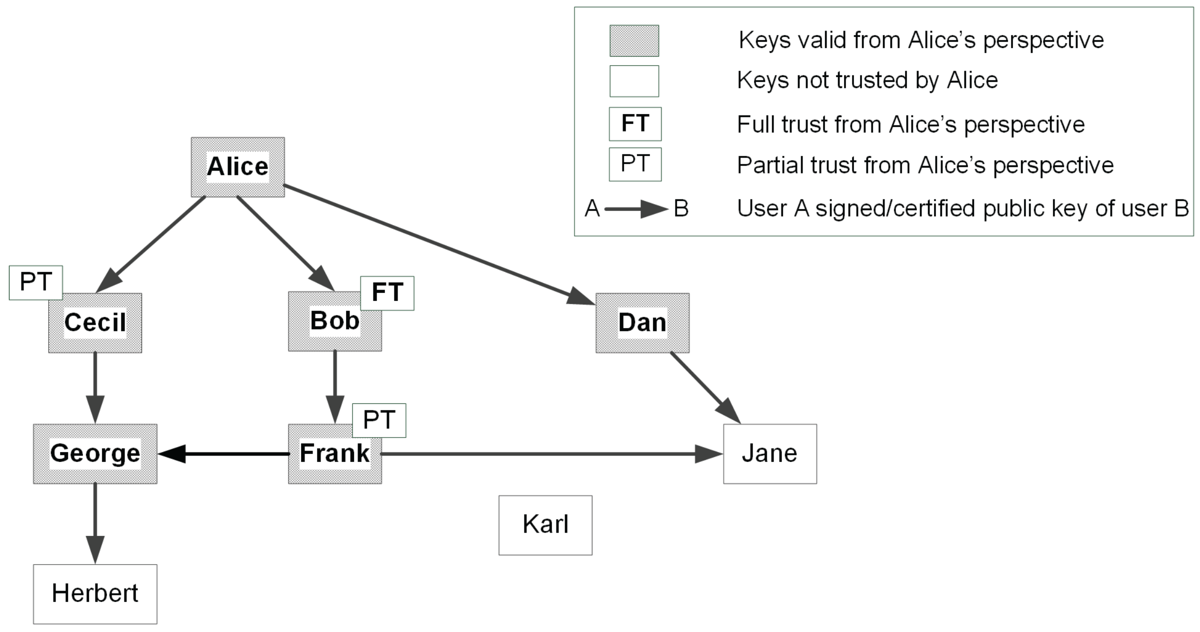
Decentralized key distribution is based on key certification. Key distribution is provided in two ways. First, users exchange public keys directly by utilizing secure channel (same way as in key continuity), but afterwards the users can digitally sign/certify other public keys (trust keys indirectly). This way it is possible to build a trust relationship and distribute public keys between users through insecure channel (send signed keys by Internet, publish them in directory/keyserver or on a web page).

Decentralized key distribution is used for example in popular mail encryption standard OpenPGP. OpenPGP designers (Zimmermann & Callas, 2009) define trust as follows: "Trust is the mechanism we use to decide that a key is valid. A key is *valid* if it is actually owned by the person who claims to own it. Validity is a score and can be used only to determine whether the name on a key is accurate. Trust is a relationship that helps us determine validity."

OpenPGP trust model (Web-of-Trust) provides two levels of trust for every entity: fully trusted and partially trusted. Valid keys are keys either: directly trusted (signed by the verifier), or signed by entity that is fully trusted by verifier, or signed by at least two entities that are at least partially trusted by verifier. This concept is illustrated by following example and Figure 3.

In example illustrated by Figure 3 Alice signed three keys: Cecil's, Bob's and Dan's. Each of these keys is directly trusted and therefore valid from Alice's perspective. Moreover Alice fully trusts

Figure 3. OpenPGP trust model (Web-of-trust) illustration


Keys valid from Alice's perspective
Keys not trusted by Alice
FT Full trust from Alice's perspective
PT Partial trust from Alice's perspective
A → B User A signed/certified public key of user B
Alice
Cecil
PT
Bob
FT
Dan
George
Frank
PT
Jane
Karl
Herbert

Bob for signing/certifying other public keys and partially trusts Cecil for signing/certifying other public keys.

Frank's key is valid due the fact that it is certified by Bob, furthermore Alice decided she will partially trust Frank as well. Finally George's public key is valid from Alice's perspective as it is signed by two partially trusted entities: Cecil and Frank.

On the other side Karl's key is not valid as there is no connection between Karl and Alice. Also even though George signed Herbert's key, it is not valid from Alice's perspective as Alice does not trust George for key certification. Finally Jane's key is not valid from Alice's perspective because it is signed only by one partially trusted entity (Frank). Dan signed it as well but he is not trusted for key certification by Alice.

In decentralized key distribution infrastructures it is possible to perform trust revocation. For example Web-of-Trust model offers two mechanisms for revocation: key revocation and signature/certification revocation. The first mechanism is useful if Alice lost her keys, or her private key is compromised. The latter mechanism is used if Alice was tricked into signing for example Dan's key.

Signature/certification revocation is performed in the same way as the signature creation. If Alice decides she was tricked into signing Dan's key, she will sign statement asserting the certification was invalid.

The key revocation in Web-of-Trust model is performed via revocation certificate. Revocation certificate is a statement asserting the key is no longer valid. This certificate should be signed by the key that needs to be revoked. Thus revocation certificate should be generated immediately after generating key pair in pre-operational phase (see Figure 1) and stored in a secure place. If the user's private key is lost or compromised, the user needs to publish this certificate and notify other users in order to revoke key.

Though the possibility of trust revocation in decentralized key distribution infrastructure is a huge improvement, it still suffers from several issues (Zimmermann & Callas, 2009). The most notable drawback is lack of simple mechanism for distribution of revocation information. The revocation information is distributed in decentralized key distribution infrastructure by same means as cryptographic keys. This means that either one needs to distribute the revocation certificate to everyone that could need it, or publish it to a central directory (keyserver).

According to (Zimmermann & Callas, 2009) Web-of-Trust scales at the low end up to few thousand people. Moreover if people share a single community, enterprise or organization, it scales even up tens of thousands of people. This estimate was empirically proven to be correct by a study (Ulrich, Holz, Hauck, & Carle, 2011). The study examined Web-of-trust Internet infrastructure by analysing existing keys and certifications distributed through public keyservers. The study covered almost 3,000,000 keys and found that largest strongly connected component in the Web-of-Trust graph projection consists of about 45,000 nodes. On the other hand, majority of strongly connected components consisted only of about 10-100 nodes. The study concludes that Web-of-Trust is efective public key distribution infrastructure within smaller node neighborhoods, and particularly for those users that frequently sign other keys and are active in the Web-of-Trust.

Unfortunately several downsides remain for decentralized key distribution as well. First, the decentralized key distribution has low support in most of European Union country legislations for legally binding electronic transactions. These are mostly supported by electronic signatures built upon a centralized hierarchical key distribution infrastructure. Second, decentralized key distribution, especially Web-of-Trust, has complicated and hard to understand trust model (Zurko & Simon, 1996), (Abdul-Rahman, 1997). Finally, this also affects the usability of Web-of-Trust, which is in fact in a bad shape (Whitten & Tygar, 1999), (Kapadia, 2007).

Flat Centralized Key Distribution

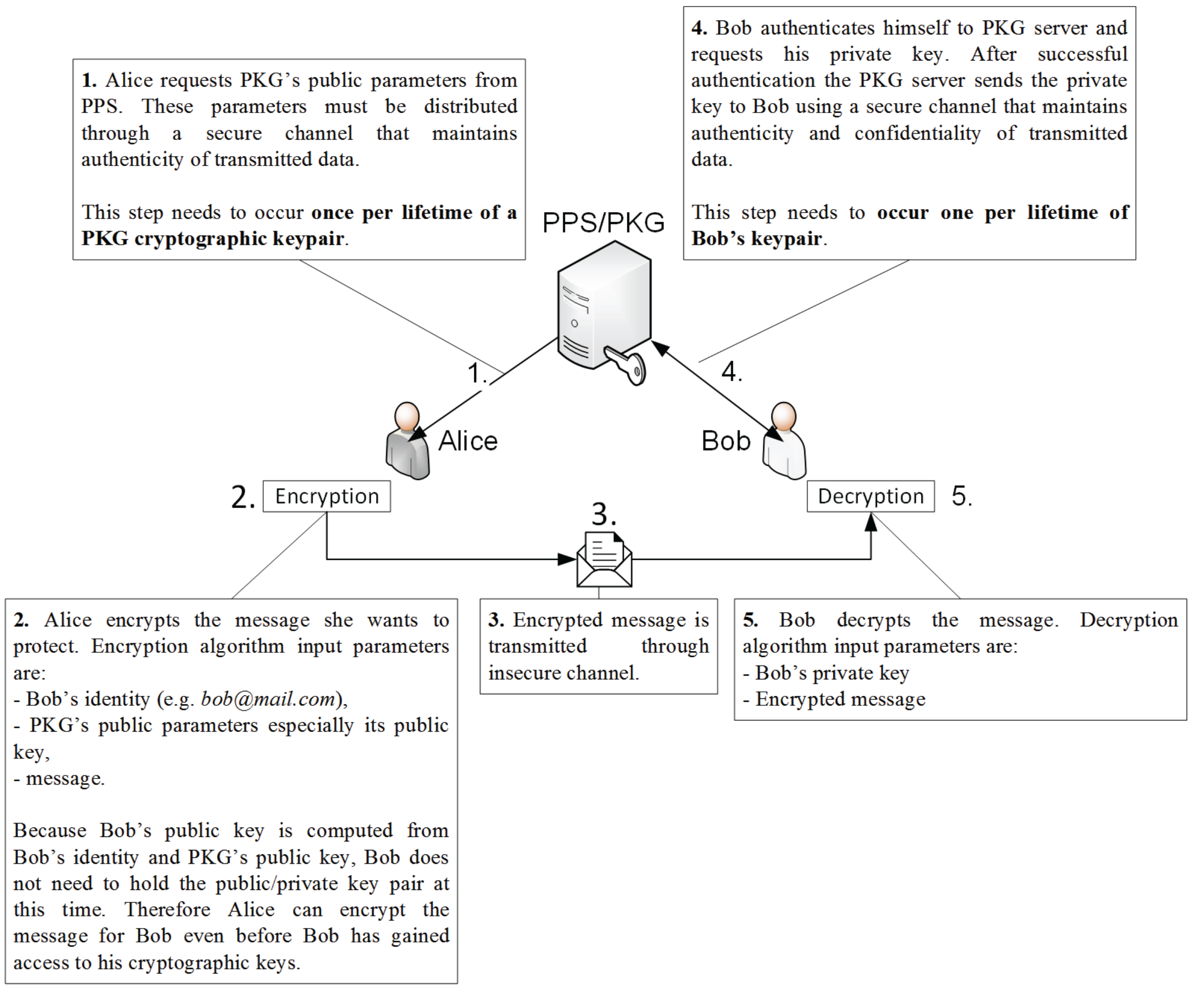
In flat centralized public key distribution infrastructure significant amount of key management phases is under supervision of a single central authority. This central authority is usually responsible for generating the cryptographic key pairs, binding the public key and its holder's identity, and for the overall key distribution process.

Flat centralized key distribution approach is mostly used in conjunction with identity based encryption[2]. *Identity based encryption* is a public-key encryption technology that allows a user to calculate a public key from an arbitrary string (Martin L., 2008). In identity based encryption the holder's public key can be directly derived from holder's identity (such as e-mail address, personal number, IP address, or DNS name). Consequently, identity based encryption should enable intuitive and user friendly approach to key distribution (Martin L., 2008).

Identity based encryption is built upon a central authority called *private key generator* (Baek, Newmarch, Safavi-Naini, & Susilo, 2004). The private key generator has its own cryptographic key pair. This key pair is used within key distribution infrastructure for generating other participants' key pairs. The public key of a private key generator is used for construction of participant's public keys. Its corresponding private key is used for generating participant's private keys. This principle is demonstrated by Figure 4 which is described below as follows.

Let us assume Alice wants to send encrypted message to Bob. Alice and Bob need to trust joint private key generator (PKG). From Alice's point of view this means that she has obtained PKG's public parameters especially PKG's public key, applicable cryptographic algorithms and other public information (Appenzeller, Martin, & Schertler,

Figure 4. IBE example


1. Alice requests PKG's public parameters from PPS. These parameters must be distributed through a secure channel that maintains authenticity of transmitted data.

This step needs to occur **once per lifetime of a PKG cryptographic keypair**.
PPS/PKG
4. Bob authenticates himself to PKG server and requests his private key. After successful authentication the PKG server sends the private key to Bob using a secure channel that maintains authenticity and confidentiality of transmitted data.

This step needs to **occur one per lifetime of Bob's keypair**.
1.
4.
Alice
Bob
2. Encryption
3.
Decryption 5.
2. Alice encrypts the message she wants to protect. Encryption algorithm input parameters are:
- Bob's identity (e.g. *bob@mail.com*),
- PKG's public parameters especially its public key,
- message.

Because Bob's public key is computed from Bob's identity and PKG's public key, Bob does not need to hold the public/private key pair at this time. Therefore Alice can encrypt the message for Bob even before Bob has gained access to his cryptographic keys.
3. Encrypted message is transmitted through insecure channel.
5. Bob decrypts the message. Decryption algorithm input parameters are:
- Bob's private key
- Encrypted message

2009). Alice can obtain these parameters either from a dedicated *public parameter server* (PPS) or directly from PKG. After completing the initial trust setup Alice can use PKG's public key and Bob's identity in order to encrypt messages for Bob. To decrypt a message Bob needs to authenticate himself to the PKG and prove his identity. Subsequently, Bob obtains his private key which is used decrypt the message.

Besides more intuitive key and identity binding mechanism, identity based encryption is also attractive because of small communication overhead. Alice needs to obtain PKG's public parameters at least once per its key pair lifetime. Likewise Bob needs to communicate with PKG only when he needs to obtain his private key. In theory, this could be limited also once per Bob's key pair lifetime.

Identity based encryption provides trust revocation by using short-term cryptographic keys. Bob's key lifetime can be limited by combining Bob's identity and his key pair validity time interval (Martin & Schertler, 2009). In such case Bob's identity in encryption algorithm parameters is modified to *bob@mail.com*||*not-before-date*. This way is Bob forced to request new key from PKG each time new *not-before-date* is specified.

This type of revocation mechanism is preventive; revocation occurs regardless of key compromise. Thus this mechanism might be more effective in environments where key compromise is not be detected easily. On the other hand shortening Bob's key pair lifetime increases the communication overhead. Especially the number of times Bob needs to authenticate to PKG are increased. Consequently secure channel that maintains authenticity and confidentiality of transmitted data between PKG and Bob needs to be set-up more frequently.

Another noteworthy feature of identity based encryption is built-in key escrow. Because Bob's private key is generated by PKG and not directly by Bob, PKG can reveal this key to Bob or other (authorized) entity at any time. This significantly simplifies key backup and restore from the user's point of view, but also poses a threat of impersonation. If an attacker is able to authenticate to PKG as Bob, the attacker can obtain any private key issued to Bob up to current date (Martin & Schertler, 2009).

Support for non-repudiation and authentication services in identity based encryption is arguable due to key escrow feature. Though identity based signature schemes exists (Shamir, 1985), (Hess, 2003), (Joye & Neven, 2009), it is believed that identity based encryption should not provide authentication and non-repudiation services (Martin L., 2008). This presumption is also supported by the fact that identity based signatures are not deployed in practice.

In real-world flat centralized key distribution infrastructure and identity based encryption is mostly used in enterprise environments for encryption of data, especially for e-mail protection. Notable implementations include Microsoft Exchange Hosted Protection and implementations by Voltage Security. These implementations combine identity based encryption for end-to-end encryption of data between users and SSL/TLS protocol for protecting communications between users and PKG/PPS.

There were also attempts to implement and standardize the usage of identity based encryption in SSL/TLS protocol (Roschke, Ibraimi, Cheng, & Meinel, 2010), (Huang, 2009). These attempts were unsuccessful, because flat centralized key distribution suffers from similar problems as hierarchical centralized key distribution which is utilized by SSL/TLS protocol (Rescorla, 2008), (Rescorla, 2009).

Hierarchical Centralized Key Distribution

Hierarchical centralized key distribution is based on key certification in a similar way as decentralized key distribution. The main difference resides in responsibility for key certification. In decentralized key distribution the keys can be certificated by anyone, whereas in centralized key distribution the keys are certified by dedicated entities. These authorities are responsible for binding the public key and its holder's identity.

Most known and widespread centralized key distribution infrastructure is X.509 public key infrastructure (PKI). In X.509 PKI the authorities responsible for binding the public key and its holder's identity are called *certification authorities* and the binding is realized through X.509 *certificates* signed by these authorities.

In centralized infrastructures the key distribution is de-facto reduced to key distribution of certification authorities' public keys (trust anchors, for example self-signed certificates). These keys/certificates are distributed mostly with software that is performing the cryptographic operations. These could be either applications (such as Firefox, Opera) or third party services for cryptographic key distribution (such as services provided by operating system, e.g. Microsoft Windows Root Certificate Program or Apple Root Certificate Program). The software that performs the cryptographic operations needs to be trusted and distributed through a secure channel that maintains authenticity of transmitted data. This means that centralized cryptographic key distribution does not introduce new requirements related to providing secure authenticated channel.

Certification authorities (CA) either issue certificates directly to the users (also called end entities) or to other certification authorities. The reason for having a hierarchy of certification authorities is to simplify their key/certificate distribution. This way it is possible to build an infrastructure, where only a small number of *root* certification authorities issue certificates to many subordinate certification authorities, thus many fewer root CA certificates need to be distributed. With multiple subordinate certification authorities every certification authority can have its own scope of issuing certificates. This means an own set of policies that place requirements on operational procedures, security precautions, and procedures to verify the identities of end entities.

In theory multiple subordinate certification authorities should provide a flexible and scalable hierarchical key distribution infrastructure that can be constrained by the scope of the issued certificates. This scope can be defined in several ways. One way to limit the scope is to limit the purpose of the certificate, or more specifically, to limit its usage and application (e.g. secure mail, electronic and qualified electronic signatures, network and transport layer encryption etc.). Another way to limit the scope is policy constraint. Policy constraints incorporate different security and identity verification procedures or reflect different amount of resources invested into private key protection (e.g. private keys are protected by smart cards or hardware security modules). Finally the scope can be controlled by utilizing naming constraints. If a certification authority is limited by name constraints, it can issue certificates only with specific name structure. For example such certification authority might issue only certificates for end entities located in a specific country or certificates for a specific DNS domain.

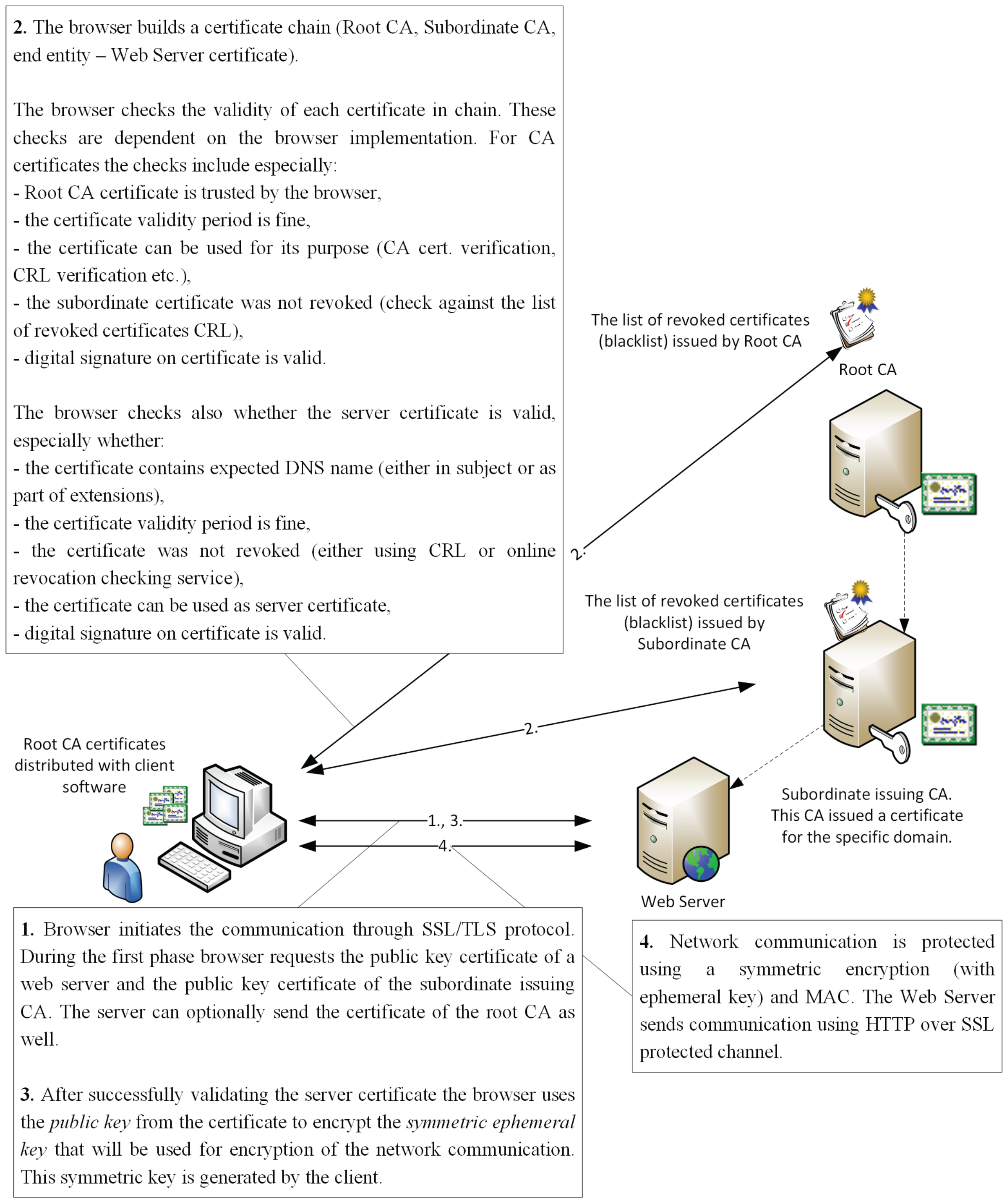
Theoretically X.509 public key infrastructure provides an elegant solution for trust revocation. Certification authorities provide revocation service, either by publishing a black-list of untrusted certificates (*certificate revocation lists*), or by providing an online service for determining the validity of issued certificates. The location of files in former case or services in latter case are stored in X.509 certificate. Relying parties that use the certificate use these information to check if the certificate has not been revoked.

X.509 certificates are widely used for many purposes. X.509 public key infrastructure is used for distribution and managing of keys used for network level encryption (SSL/TLS, IPSec) as well as application level encryption, signing and authentication (S/MIME, electronic signatures, code signing, etc.). To explain the basic points of key certification, distribution and revocation in X.509 we choose SSL/TLS example illustrated by Figure 5.

Unfortunately even though X.509 PKI looks appealing at first sight, due to its complexity and flexibility it is hard to implement in real-world in an efficient and secure way (Gutmann, 2002). Especially SSL/TLS PKI is known to be in poor condition due to the prevalence of certificate errors, improper configuration and administrator's mistakes (Holz, Braun, Kammenhuber, & Carle, 2011), and too many certification authorities with poor security practices (Peter & Burns, 2010), (Arnbak & Van Eijk, 2012).

There were several actual attacks against SSL/TLS PKI. In 2011 two widely trusted certification authorities: Comodo CA and DigiNotar, were breached. During the security breach fraudulent certificates were issued to an attacker that did not control the specific DNS domain names (Leavitt, 2011). This way the attacker could mount a successful attack against any web site that used SSL/TLS protocol communications encryption as long as he could control the network on lower layers. There are indications that these fraudulent certificates were used during an Iranian attack on public e-mail services such as Google's Gmail (Arnbak & Van Eijk, 2012).

Figure 5. SSL/TLS example


The list of revoked certificates (blacklist) issued by Root CA
Root CA
The list of revoked certificates (blacklist) issued by Subordinate CA
Root CA certificates distributed with client software
2.
2.
1., 3.
4.
Subordinate issuing CA. This CA issued a certificate for the specific domain.
Web Server

2. The browser builds a certificate chain (Root CA, Subordinate CA, end entity – Web Server certificate).

The browser checks the validity of each certificate in chain. These checks are dependent on the browser implementation. For CA certificates the checks include especially:
- Root CA certificate is trusted by the browser,
- the certificate validity period is fine,
- the certificate can be used for its purpose (CA cert. verification, CRL verification etc.),
- the subordinate certificate was not revoked (check against the list of revoked certificates CRL),
- digital signature on certificate is valid.

The browser checks also whether the server certificate is valid, especially whether:
- the certificate contains expected DNS name (either in subject or as part of extensions),
- the certificate validity period is fine,
- the certificate was not revoked (either using CRL or online revocation checking service),
- the certificate can be used as server certificate,
- digital signature on certificate is valid.

1. Browser initiates the communication through SSL/TLS protocol. During the first phase browser requests the public key certificate of a web server and the public key certificate of the subordinate issuing CA. The server can optionally send the certificate of the root CA as well.

3. After successfully validating the server certificate the browser uses the *public key* from the certificate to encrypt the *symmetric ephemeral key* that will be used for encryption of the network communication. This symmetric key is generated by the client.

4. Network communication is protected using a symmetric encryption (with ephemeral key) and MAC. The Web Server sends communication using HTTP over SSL protected channel.

The attack is quite simple. During the first phase of a SSL/TLS protocol handshake, the attacker performs a man in the middle attack by inserting a legitimate certificate from compromised certification authority. If the certification authority is not aware of the compromise, or if the attacker can block CRL/OCSP communication with CA, then the browser cannot determine whether the certificate was fraudulently issued and considers the certificate valid. Following this, the client will encrypt the symmetric key with attacker's public key and attacker can decrypt all the traffic designated for the web server. He can than act as an active proxy server in the communication.

The reasons why this attack is relatively easy can be summarized as follows. Revocation either by certificate revocation lists or online validation services is by default fail-safe; meaning that if the browser is unable to retrieve revocation information the SSL/TLS channel setup proceeds, and sometimes it proceeds even without a warning (TrustWave SpiderLabs, 2011). As mentioned earlier there are many (perhaps too many) certification authorities trusted by the internet browsers. Therefore it is easier for the attacker to find a less secured but equally trusted certification authority. Also with so many certification authorities it is hard for an average user to determine which are trustworthy and which are not (Schultze, 2010).

Furthermore the attack can be even less sophisticated and still successful. The attacker can issue the web server certificate by a self-crafted certification authority that is not trusted by the browser. Most of the internet browsers will show a warning that the certificate is issued by an untrusted certification authority and recommend the user to abandon the page. Unfortunately significant amount users ignore this warning, thus the attack is successful. This suggestion is backed up by a study performed on more than 400 users (Sunshine, Egelman, Almuhimedi, Atri, & Cranor, 2009). This study showed that more than 30% of users ignored warnings related to untrusted certification authorities.

Several attacks of this nature were detected on the Internet. Among others it is worthy to mention that in 2011 Facebook has been targeted by this type of attack (Eckersley, 2011) and in 2013 popular source code repository Github was attacked as well (CloudShark, 2013).

But as we stated earlier, it is hard to blame the users for ignoring browser warnings, when the TLS/SSL PKI is in a bad shape. In a thorough study covering SSL/TLS certificates from Alexa Top 1 Million Hosts (Holz, Braun, Kammenhuber, & Carle, 2011) it has been shown that only around 18% of certificates were valid with regards to checking rules implemented in popular browsers. The remaining certificates could not be used for securing TLS/SSL channel unless user acknowledged a browser warning.

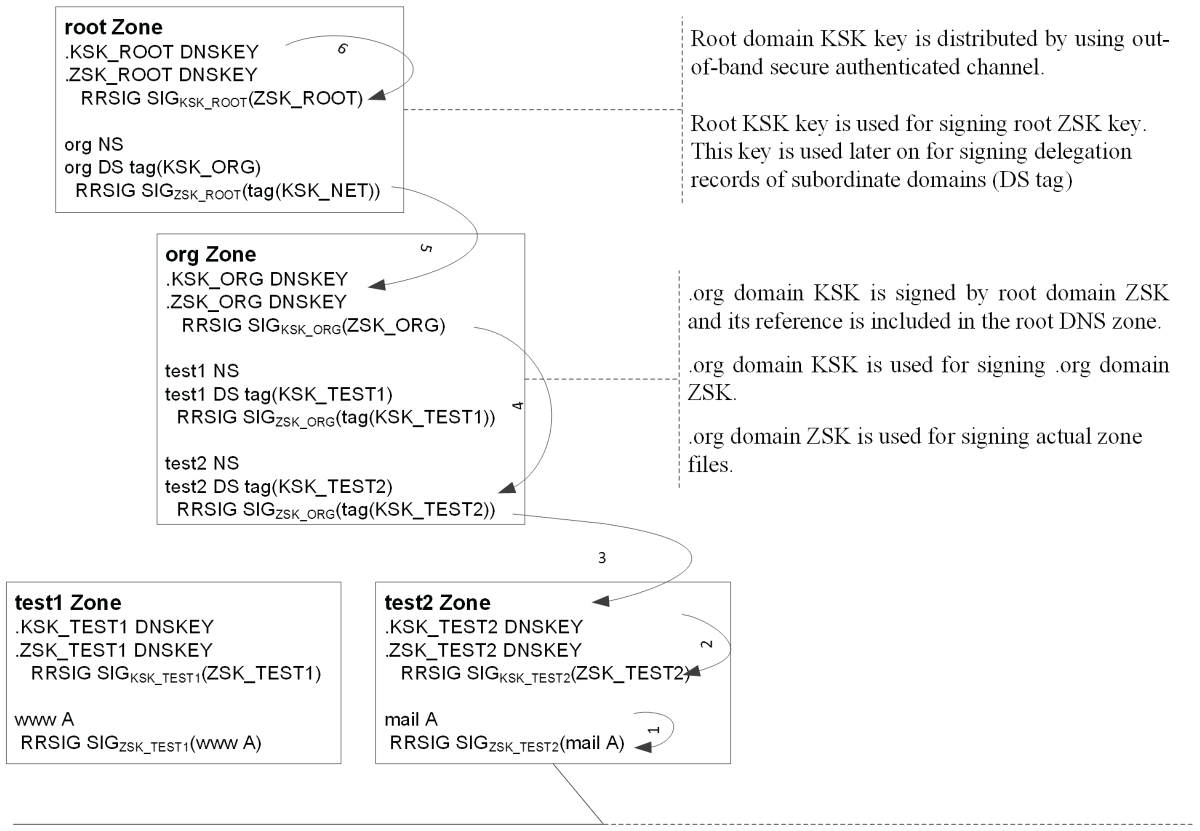
A more recent example of hierarchical key distribution infrastructure is DNSSEC PKI. DNSSEC PKI does not suffer from some problems of traditional X.509 PKI. DNSSEC PKI is focused only on distribution and management of keys relevant for securing the DNS, and constrains certification authorities out-of-box, thus it is more focused and constrained than traditional X.509 PKI.

DNSSEC makes use of two types of keys: zone signing key (ZSK) and key signing key (KSK). ZSK is used for signing actual DNS resource records (RR) and KSK is used for signing ZSK. This way ZSK can be rolled over more often, thus it needs to fulfill less stringent security requirements (such as requirements on key size or operational security) than KSK.

Public key part of ZSK is published along with public key part of KSK in domain. In order to provide chain-of-trust public key part of KSK (its identifier tag- hash) is published as a signed record in the parent DNS domain. The chain of trust is illustrated by Figure 6.

DNSSEC PKI does not suffer from the over-flexibility of X.509 PKI. Unfortunately it still too complex, especially with regards to deployment and overall infrastructure operations related to DNS. A study covering aspects of large-scale

Figure 6. DNSSEC chain of trust example


root Zone
.KSK_ROOT DNSKEY
.ZSK_ROOT DNSKEY
RRSIG SIG$_{KSK\_ROOT}$(ZSK_ROOT)
org NS
org DS tag(KSK_ORG)
RRSIG SIG$_{ZSK\_ROOT}$(tag(KSK_NET))
org Zone
.KSK_ORG DNSKEY
.ZSK_ORG DNSKEY
RRSIG SIG$_{KSK\_ORG}$(ZSK_ORG)
test1 NS
test1 DS tag(KSK_TEST1)
RRSIG SIG$_{ZSK\_ORG}$(tag(KSK_TEST1))
test2 NS
test2 DS tag(KSK_TEST2)
RRSIG SIG$_{ZSK\_ORG}$(tag(KSK_TEST2))
test1 Zone
.KSK_TEST1 DNSKEY
.ZSK_TEST1 DNSKEY
RRSIG SIG$_{KSK\_TEST1}$(ZSK_TEST1)
www A
RRSIG SIG$_{ZSK\_TEST1}$(www A)
test2 Zone
.KSK_TEST2 DNSKEY
.ZSK_TEST2 DNSKEY
RRSIG SIG$_{KSK\_TEST2}$(ZSK_TEST2)
mail A
RRSIG SIG$_{ZSK\_TEST2}$(mail A)
1
2
3
4
5
6

Root domain KSK key is distributed by using out-of-band secure authenticated channel.

Root KSK key is used for signing root ZSK key. This key is used later on for signing delegation records of subordinate domains (DS tag)

.org domain KSK is signed by root domain ZSK and its reference is included in the root DNS zone.

.org domain KSK is used for signing .org domain ZSK.

.org domain ZSK is used for signing actual zone files.

Example: mail.test2.org A DNS record should be verified.

1. mail.test2.org DNS record signature is verified by test2 zone signing key (ZSK_TEST2)
2. ZSK_TEST2 DNS record signature is verified by test2 key signing key (KSK_TEST2)
3. KSK_TEST2 signed tag (key identifier) is published in parent domain (org) and verified by org zone signing key (ZSK_ORG)
4. ZSK_ORG DNS record signature is verified by org key signing key (KSK_ORG)
5. KSK_ORG signed tag (key identifier) is published in root domain and verified by root zone signing key (ZSK_ROOT)
6. ZSK_ROOT DNS record signature is verified by root key signing key (KSK_ROOT) which is distributed by out-of-band secure authenticated channel

Internet deployment of DNSSEC (Yang, 2011) shows that several issues related to X.509 PKI operations apply also to DNSSEC PKI. Moreover new problems arise because of large-scale delegation of certification authorities.

As (Yang, 2011) states the fundamental challenge in operating DNSSEC is that the system spans over tens of millions of independent administrative domains, while the provision of security, as defined in DNSSEC, requires symphonic actions from all of them. As the public keys of different zones are authenticated through a single hierarchical PKI, any local change in a zone's public key may require synchronization across administrative domains, because the new key must be authenticated by the zone's parent and used to authenticate the zone's children. This causes serious scaling issues for the domains that may have millions of children zones underneath them. The coordination process is also lengthy and error prone as it involves human operators, yet any out-of-sync configurations can break the chains of trust and disrupt the DNS service due to authentication failure.

These issues and the fact that DNSSEC does not improve network security significantly unless it is deployed wide enough (Friedman, 2011) lead to slow adoption of DNSSEC. Though the deployment trend is steadily rising over six years (Verisign Labs, 2013), its overall coverage is presumed to be less than 2% of second level DNS domains (Eggert, 2013). To be more specific less than 1% of *.edu* DNS domains are DNSSEC enabled and less than 0.3% of *.net* and *.com* DNS domains are DNSSEC enabled. On the other hand Czech second level domain is a bright example of how DNSSEC penetration can be achieved. Actual DNSSEC penetration in *.cz* DNS domain is about 38% (cz.nic, 2013), and has been achieved by great CZ.NIC effort especially in terms of marketing, smart economic policy, trainings, users and administrators education (Filip, 2011).

COMPARISON OF PUBLIC KEY DISTRIBUTION APPROACHES

We summarize existing approaches and deployed public key distribution and management infrastructures in tables below.

Table 2 illustrates core characteristics of public key distribution approaches, Table 3 compares their practical aspects, and Table 4 outlines overall advantages and disadvantages of a particular key distribution approach.

Table 2. Public key distribution approaches- Core characteristics

Key Distribution Approach Characteristics	**Key Continuity**	**Decentralized Key Distribution**
	Flat centralized key distribution	**Hierarchical centralized key distribution*** * specific aspects for particular PKI implementation are amended respectively (e.g. X.509 or DNSSEC)
Secure (Authenticated) Channel Requirements	**K** Each time new public key binding is set up or modified.	**D** Each time when direct binding is setup.
	F Once for distribution of PKG's public parameters. Moreover secure channel that preserves authenticity and **confidentiality** of transmitted data is required every time an entity requests private key.	**H** Theoretically once; during the distribution of trusted third parties certificates/trust anchors.
Trust Options (Direct / Indirect)	**K** Public key bindings must be verified directly.	**D** Public key bindings are verified either directly or indirectly by trusting other participants of the infrastructure.
	F Public key bindings are established through identity directly. Private key is obtained by its holder after successful authentication to PKG.	**H** Public key bindings are established / verified by trusted third party.
Trust Revocation	**K** Impossible without authenticated channel. Everyone (who makes use of the public key) needs to be notified about the trust revocation through an authenticated channel.	**D** Theoretically possible, not much used in practice. Everyone (who makes use of the public key) should check key servers for trust revocation.
	F Possible via short-term keys. Shortening the end entity key validity period increases communication overhead and might introduce administrative and operational issues.	**H-X.509** Everyone (who makes use of the public key) should check certification authority services for trust revocation. **H-DNSSEC** Possible via short-term keys. DNSSEC key revocation needs to consider the DNS caching mechanisms.

Table 3. Public key distribution approaches - practical aspects

Key Distribution Approach Characteristics	**Key Continuity**	**Decentralized Key Distribution**
	Flat centralized key distribution	**Hierarchical centralized key distribution*** * specific aspects for particular PKI implementation are amended respectively (e.g. X.509 or DNSSEC)
Practical implementations	**K** SSH, ZRTP	**D** PGP
	F IBE: RFC 5408, RFC 5409	**H** X.509, DNSSEC
Scalability	**K** Low ~ 10-100 keys - parties	**D** Medium ~ 10.000 keys - parties / Enterprise wide
	F Medium ~ Enterprise wide	**H** High ~ Internet wide
Usability in existing applications	**K** Hard to use for common users, Easy to configure for administrators.	**D** Hard to use for common users.
	F Intuitive for common users, easy to configure for administrators.	**H** Hard to use for common users, hard to configure and operate for administrators.
Estimated infrastructure costs[3]	**K** Low Set-up: Up to 15 minutes *per key* ~ 15 USD	**D** Medium Set-up: At least an hour *per key*/user ~ 60 USD
	F Medium Set-up: 30,000 USD Operational: Up to 10 USD per user/month (enterprise environment)	**H** High - Aproximately 100,000 USD for enterprise certification authority Operational: Up to 20 USD per user/month (enterprise environment)

REINFORCING THE EXISTING (TLS/SSL) X.509 PUBLIC KEY INFRASTRUCUTRE

As we stated earlier, several incidents indicate that SSL/TLS PKI is prone to mis-issuance of X.509 certificates. According to (Schultze, 2010) the fundamental weaknesses reside in the fact that any CA can issue a certificate vouching that the subscriber controls any domain name, accurate or not. This is due missing/unimplemented constraints in X.509 trust model and unclear/undisclosed delegation of trust to other certification authorities by root certification authorities.

The DNS-Based Authentication of Named Entities (DANE)

Two reinforcement solutions for X.509 SSL/TLS PKI trust model were standardized recently. Both of them make use of DNS infrastructure in order to provide additional binding to the public keys and their holders' identity and provide a way to constrain the trust.

Last year IETF DANE Working Group published RFC 6698. The purpose of this RFC is to introduce a new DNS resource record TLSA, and provide additional security measure for existing SSL/TLS PKI.

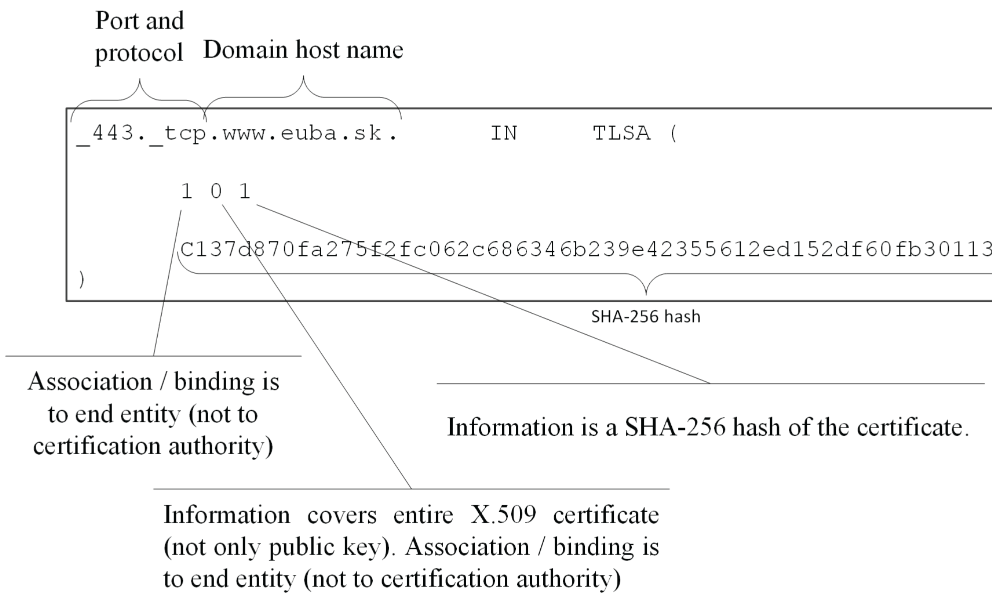
TLSA DNS resource record binds TLS port, DNS name and holder's public key association. The public key association can provide either X.509 certificate or raw public key, or its hash. Also the binding can be direct (subject's public key or X.509 certificate) or indirect (issuing certification authority public key or X.509 certificate). An example of such association is provided by Figure 7.

Because DNS is not secure by default, DANE relies on DNSSEC in order to provide authenticated channel for the associations. This way the traditional X.509 public key infrastructure can be combined along with DNSSEC PKI. By combining DNSSEC PKI and X.509 PKI the attacker would need to subvert a regular certification authority in order to issue a false certificate and compromise the target's DNSSEC infrastructure as well.

Table 4. Public key distribution approaches - overall advantages and disadvantages

Key Distribution Approach Characteristics	**Key Continuity**	**Decentralized Key Distribution**
	Flat centralized key distribution	**Hierarchical centralized key distribution*** *- specific aspects for particular PKI implementation are amended respectively (e.g. X.509 or DNSSEC)
Advantages	**K** - Simple, inexpensive, and easy to setup and operate	**D** - Scales well for medium size environments - Flexible trust model
	F - Simple and easy to setup - Intuitive public keys - Built-in escrow (from operational and compliance point of view)	**H** - Scales well for internet wide infrastructures. **H-X.509** - Overall flexibility - Possible to use for legally binding operations **H-DNSSEC** - Hierarchy is inherently constrained by DNS namespace
Disadvantages and concerns	**K** - Missing revocation - Low scalability	**D** - Complicated trust model - Complicated revocation
	F - Every time entity requests its private key a secure channel must to be established. - Key holders need to authenticate to PKG by other means than provided by PKG. This means that key holders need shared secret such as password, X.509 certificate, or cryptographic key. - Built-in escrow (from privacy and non-repudiation point of view)	**H** - Costly to set up and operate - Too complicated, hard to implement correctly **H-X.509** - Constraints on the hierarchy are not implemented in practice - Revocation does not work in practice - Mis-issuance of X.509 certificates happens in practice - Though X.509 hierarchy should be used for constraining trust, it is mostly used for profit **H-DNSSEC** - Slow adoption - Error prone certification process, with complicated trust relationships (registry, registrar, registrant, zone operator)

Figure 7. Example of DANE TLSA RR



```
_443._tcp.www.euba.sk.      IN     TLSA (

       1 0 1

       C137d870fa275f2fc062c686346b239e42355612ed152df60fb30113461406d6
)
```

Port and protocol
Domain host name
SHA-256 hash
Association / binding is to end entity (not to certification authority)
Information is a SHA-256 hash of the certificate.
Information covers entire X.509 certificate (not only public key). Association / binding is to end entity (not to certification authority)

In theory DANE should significantly reduce attack surface for the attacker (Osterweil, Kaliski, Larson, & McPherson, 2012). Without DANE protection the attacker could compromise any certification authority in order to launch a man in the middle attack. If DANE protection is applied the attacker needs to compromise the DNSSEC infrastructure, with much less applicable certification authorities (Osterweil, Kaliski, Larson, & McPherson, 2012).

The attack surface for traditional X.509 SSL/TLS PKI is estimated to at least 150 targets. The DANE reduces this attack surface (Osterweil, Kaliski, Larson, & McPherson, 2012) to the number of authoritative name servers potentially involved in resolving a DNS name. This process involves the name servers from the root DNS zone until the zone that contains the particular DNS name. The precise number of name servers varies depending on a particular domain, but in general it is by order of magnitude smaller than the number of targets in X.509 SSL/TLS PKI. A practical survey (Ramasubramanian & Sirer, 2005) of 593,160 unique webserver names crawled by the Yahoo! and DMOZ.org directories shows, that median number of the names servers in the chain is 26 and the average number of the name servers in the chain is 44.

Moreover as illustrated by Figure 7, TLSA RR can bear all the information necessary for creating a secure SSL/TLS channel. Thus DANE can be used for public key distribution without the necessity for X.509 certification authorities.

Unfortunately, as we stated before, the DNSSEC is not widely deployed and we shall see if DANE could be one of the deployment drivers. Also with DANE deployed, one would need to incorporate the trust and public key distribution changes into user interface and other browsers' components. Finally, as shown by studies (Holz, Braun, Kammenhuber, & Carle, 2011) and (Yang, 2011), it is hard to believe that DNSSEC and DANE could be applied in a large scale in an error proof way.

DNS Certification Authority Authorization

A similar approach to DANE was chosen by the PKIX IETF working group. RFC 6844 describes the Certification Authority Authorization (CAA) DNS resource record. CAA DNS RR allows a DNS domain holder to specify one or more certification authorities that are authorized to issue certificates for the domain.

The resource record could contain information about a certification authority that is allowed to issue a certificate for a specific domain, as well as contact information that could be used by a certification authority in cases, where the CA is unauthorized to issue a certificate (based on CAA RR). In this way the domain owner could be notified when someone is trying to acquire a certificate from another CA than the one that is indicated in CAA RR.

The main difference between CAA and DANE RR is that CAA RR is designated for certification authorities whereas DANE is designated for client processing. Also CAA RR cannot be used for secure key distribution. It is intended only as an additional cross-check condition in certificate vetting and issuance process.

Because the CAA cannot be used for key distribution purposes, the DNSSEC protection of the information in transit is not required. Thus CAA can be deployed without DNSSEC and also without any changes to the client codebase. This means that CAA is ready for deployment de-facto immediately.

Certificate Transparency

Similarly to CAA the Certificate Transparency is a cross-checking, auditing, and monitoring countermeasure. The goal is to make it impossible (or at least very difficult) for a certification authority to issue a certificate for a domain without it being visible to the owner of that domain.

The main idea is to create a distributed logging infrastructure that should log every publicly issued TLS X.509 certificate. This would be achieved by running several servers with cryptographically secured append-only logs. Clients could verify correctness of each log and monitor when new certificates are added. Certificate Transparency clients can be divided by their role into several groups: submitters, TLS clients, monitors and auditors.

Submitters can post a certificate to log servers. Anyone can be a submitter, examples are: certification authorities, holder of public key or a relying party. The only current condition for accepting the submission is that the certificate should be issued by a globally trusted certification authority. This should prevent spamming the log.

TLS clients make use of the log by accepting the proof of submissions of a particular certificate to the log. They validate the proof of submissions by utilizing asymmetric cryptographic protocols. The current proposal of Certificate Transparency Internet-Draft does not deal with public key distribution necessary for validating the proof of submissions.

Monitors and auditors cross-check that the logs are correctly operated. Moreover monitors could check for specific certificates or domain names in interest in order to detect mis-issuance.

The Certificate Transparency is still in experimental stage but if it is approved it should be deployed incrementally. Nonetheless, the final aim is to *require* the proof of submission by the TLS client for every publicly trusted certificate issued after a certain date (Langley, 2012).

Though the direct benefit of the certificate transparency is conditioned by changing the TLS client codebase, the entire Internet TLS infrastructure would benefit if the updated TLS client codebase is deployed widely enough. The benefit is similar to benefit that brings Google Chrome public key pinning. Public key pinning is a term for hardcoded public keys and domain names into the Google Chrome internet browser. This feature helps to detect mis-issued certificates for high valued sites, especially related (but not only) to Google online services. DigiNotar breach (Leavitt, 2011) and Turktrust certificate mis-issuance (Rosa & Schultze, 2013) had been detected thanks to public key pinning. It is straightforward that after the mis-issuance is detected the MITM attack has lower chance to succeed.

Certificate Transparency could be perceived as a scalable certificate pinning solution. The fact that it is driven by Google gives a high chance that it would be implemented at least in Google Chrome when it is finished. If the Certificate Transparency would be deployed as planned in a fail-secure blocking way (Langley, 2012), it would significantly reduce the risk of social engineering, incorrect user decisions and it would also drive usability changes in the browser towards better directions.

CONCLUSION

It is easiest to use pre-shared keys and key continuity distribution approach for smaller scale systems and forget about complexities introduced by more robust key distribution approaches. If the system needs to scale more, or revocation capabilities are needed, it is necessary to pick either a decentralized or centralized public key distribution approach.

Decentralized public key distribution is best suited for small enterprise environments and medium sized closed communities. In these environments decentralized public distribution provides a cost effective, but not very user friendly way to protect data confidentiality, authentication and non-repudiation.

Flat centralized key distribution and identity based encryption is used mostly in enterprise environments, where supervised key escrow is considered desirable feature. Identity based encryption provides user friendly and cost effective way to protect data confidentiality, but lacks support for authentication and non-repudiation in real-world implementations.

Hierarchical key distribution supports wide range of security services including: authentication, data confidentiality and non-repudiation. Hence hierarchical key distribution, especially X.509 PKI, is often implemented in enterprise environment for email protection, encryption of network communications, authentication, and digital and electronic signatures. Hierarchical key distribution is also most deployed solution internet-wide and X.509 hierarchical PKI is defacto the only real-world implemented approach for managing SSL/TLS public keys on the Internet.

Though X.509 PKI approach has severe trust issues, it will probably stay the most widespread Internet public key infrastructure for a while. The solutions for the X.509 PKI trust problems are hard to deploy, especially when it is necessary to update client codebase, or another infrastructure needs to be deployed in a large scale.

Therefore, as short term solution CAA is most appealing, as it needs to be accepted only by certification authorities and certificate holders. In the long run Certificate Transparency is designed to provide the monitoring and detection of fraudulently issued certificates, and DANE with DNSSEC could extend the possibilities of key distribution and provide cross-checks as well.

Most of the key distribution approaches as implemented by software today, suffer from poor usability and the possibility to override security by the user's decision. Until hard fails on key distribution errors are implemented, the users will be bypassing security warnings and putting themselves to risk. But in order to implement hard fails, the infrastructure needs to be fixed first.

REFERENCES

Abdul-Rahman, A. (1997). The pgp trust model. EDI-Forum. *Journal of Electronic Commerce*, *10*(3), 27–31.

Abelson, H., Anderson, R., Bellovin, S. M., Benaloh, J., Blaze, M., & Diffie, W. et al. (1997). The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption. *World Wide Web Journal*, *2*(3), 241–257.

Adida, B., Hohenberger, S., & Rivest, R. L. (2005). Lightweight encryption for email. In *Proceedings of USENIX steps to reducing unwanted traffic on the internet workshop* (SRUTI) (pp. 93-99). Cambridge, UK: USENIX.

Amann, B., Vallentin, M., Hall, S., & Sommer, R. (2012). *Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. TR-12-014*. ICSI.

Appenzeller, G., Martin, L., & Schertler, M. (2009). *Identity-Based Encryption Architecture and Supporting Data Structures* (RFC 5408). Retrieved from http://tools.ietf.org/html/rfc5408

Arnbak, A., & Van Eijk, N. (2012). *Certificate Authority Collapse: Regulating Systemic Vulnerabilities in the HTTPS Value Chain*. TRPC. Retrieved from http://ssrn.com/abstract=2031409

Baek, J., Newmarch, J., Safavi-Naini, R., & Susilo, W. (2004). A Survey of Identity-Based Cryptography. In *Proceedings of Australian Unix Users Group Annual Conference* (pp. 95-102). Melbourne: AUUG.

Boldyreva, A., Vipul, G., & Virendra, K. (2008). Identity-based encryption with efficient revocation. In *Proceedings of the 15th ACM conference on Computer and communications security* (pp. 417-426). Alexandria: ACM.

Callas, J., & Gutmann, P. (2013). *How much does it cost to start a root CA?* Retrieved from http://lists.randombit.net/pipermail/cryptography/2013-January/thread.html#3575

Camp, J., Asgharpour, F., & Liu, D. (2007). Experimental Evaluations of Expert and Non-expert Computer Users' Mental Models of Security Risks. [Pittsburgh, PA: WEIS.]. *Proceedings of WEIS*, *2007*, 1–24.

Chandramouli, R., & Scott, R. (2009). Open issues in secure DNS deployment. *IEEE Security & Privacy*, *7*(5), 29–35. doi:10.1109/MSP.2009.129

Chen, L., Harrison, K., Moss, A., Soldera, D., & Smart, N. P. (2002). Certification of public keys within an identity based system. In *Proceedings of Information Security* (pp. 322–333). Springer. doi:10.1007/3-540-45811-5_25

CloudShark. (2013, January 29). *MITM Attack Capture Shared Through CloudShark*. Retrieved from http://appliance.cloudshark.org/news/cloudshark-in-the-wild/mitm-attack-capture-shared-through-cloudshark/

cz.nic. (2013, February 27). *Statistics*. Retrieved from http://www.dnssec.cz/stats/?stat_type=1&zone=2&time_step=month&from_year=2012&from_month=2&from_day=1&to_year=2013&to_month=2&to_day=26&submit=1

Denning, D. E., & Sacco, G. M. (1981). Timestamps in key distributed protocols. *Communications of the ACM*, *24*(8), 533–535. doi:10.1145/358722.358740

Eckersley, P. (2011, May 5). *A Syrian Man-In-The-Middle Attack against Facebook*. Retrieved from https://www.eff.org/deeplinks/2011/05/syrian-man-middle-against-facebook

Eggert, L. (2013, February 25). *DNSSEC Deployment Trends*. Retrieved from http://eggert.org/meter/dnssec

enisa. (2009). *The costs of DNSSEC deployment*. Heraklion: ENISA.

Ferris. (2006). The Total Cost of Ownership for Voltage Identity-Based Encryption Solutions. *San Francisco: Ferris Research.*

Filip, O. (2011). *DNSSEC.CZ*. Retrieved from http://dakar42.icann.org/bitcache/23b46cc058a604df99944c7080b98194591c8fd5?vid=28941&disposition=attachment&op=download

Friedman, A. (2011). *Economic and Policy Frameworks for Cybersecurity Risks*. Center for Technology Innovation at Brookings.

Furnell, S. (2005). Why users cannot use security. *Computers & Security*, *24*(4), 274–279. doi:10.1016/j.cose.2005.04.003

Garfinkel, S. L. (2003). Email-based identification and authentication: An alternative to PKI? *IEEE Security & Privacy*, *1*(6), 20–26. doi:10.1109/MSECP.2003.1253564

Garfinkel, S. L., & Miller, R. C. (2005). Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *Proceedings of the 2005 symposium on Usable privacy and security* (pp. 13-24). Pittsburgh, PA: ACM.

Gentry, C. (2003). Certificate-based encryption and the certificate revocation problem. In Proceedings of Advances in Cryptology - EUROCRYPT 2003 (pp. 272-293). Springer.

Gentry, C., & Silverberg, A. (2002). Hierarchical ID-Based Cryptography. [Springer.]. *Proceedings of ASIACRYPT*, *2002*, 548–566.

Gutmann, P. (2002). PKI: it's not dead, just resting. *Computer*, *35*(8), 41–49. doi:10.1109/MC.2002.1023787

Gutmann, P. (2004). Why isn't the internet secure yet, dammit. In *Proceedings of AusCERT Asia Pacific Information Technology Security Conference*. Royal Pines: AusCERT.

Gutmann, P. (2011). Do Users Verify SSH Keys?. *login, 36*(4), 35-36.

Gutmann, P. (2011, April 20). *The real cost of free certificates*. Retrieved from https://mail1.eff.org/pipermail/observatory/2011-April/000199.html

Herley, C. (2009). So Long, And No Thanks for the Externalities: The Rational Rejection of Security Advice by Users. In *Proceedings of Workshop on New security paradigms* (pp. 133-144). New York: ACM.

Herzberg, A., & Margulies, R. (2012). Training Johnny to Authenticate (Safely). *IEEE Security and Privacy*, *10*(1), 37–45. doi:10.1109/MSP.2011.129

Hess, F. (2003). Efficient identity based signature schemes based on pairings. In *Proceedings of Selected Areas in Cryptography* (pp. 310–324). Springer. doi:10.1007/3-540-36492-7_20

Holz, R., Braun, L., Kammenhuber, N., & Carle, G. (2011). The SSL landscape: a thorough analysis of the x. 509 PKI using active and passive measurements. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (pp. 427-444). ACM. Retrieved December 2011, from http://www.net.in.tum.de/fileadmin/bibtex/publications/papers/imc-pkicrawl-2.pdf

Holz, R., Riedmaier, T., Kammenhuber, N., & Carle, G. (2012). X. 509 forensics: Detecting and localising the SSL/TLS men-in-the-middle. [Springer.]. *Proceedings of Computer Security–ESORICS*, *2012*, 217–234.

Huang, M. (2009, December). *Identity-Based Encryption (IBE) Cipher Suites for Transport Layer*. Retrieved from http://tools.ietf.org/html/draft-huang-tls-ibe-00

ITU-T. (1991, August 30). *X. 800 Security Architecture for Open Systems Interconnection for CCITT applications*. Retrieved from http://www.itu.int/rec/T-REC-X.800-199103-I/en

Jääskeläinen, K. (2013). *SSH Key Management: A Gaping Hole in Your Encrypted Critical Infrastructure*. Retrieved from http://www.infosecurityproject.com/2013/Download/BO1.1_A%20Gaping%20Hole%20in%20Your%20Encrypted%20Critical%20Infrastructure.pdf

Joye, M., & Neven, G. (2009). *Identity-based cryptography* (Vol. 2). Amsterdam: IOS Press.

Kahn, D. (1996). *The Codebreakers*. New York: Simon & Schuster.

Kapadia, A. (2007). A case (study) for usability in secure email communication. *IEEE Security & Privacy*, *5*(2), 80–84. doi:10.1109/MSP.2007.25

Khurana, H. B. (2006, January). *On the risks of IBE*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.9742&rep=rep1&type=pdf

Kienzle, D. M., & Elder, M. C. (2003). Recent Worms: A Survey and Trends. In *Proceedings of the 2003 ACM workshop on Rapid malcode* (pp. 1-10). Washington, DC: ACM.

Langley, A. (2012, November 6). *Certificate Transparency*. Retrieved from http://www.imperialviolet.org/2012/11/06/certtrans.html

Langley, A. (2013, January 13). *Real World Crypto 2013*. Retrieved from http://www.imperialviolet.org/2013/01/13/rwc03.html

Leavitt, N. (2011). Internet Security under Attack: The Undermining of Digital Certificates. *Computer*, *44*(12), 17–20. doi:10.1109/MC.2011.367

Martin, L. (2008). *Introduction to identity-based encryption*. Norwood, MA: Artech House.

Martin, L., & Schertler, M. (2009, January). Using the Boneh-Franklin and Boneh-Boyen Identity-Based Encryption Algorithms with the Cryptographic Message Syntax (CMS) *(RFC 5409). Retrieved from* http://tools.ietf.org/html/rfc5409

Medvinsky, A., & Hur, M. (1999, October). *Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)* (RFC 2712). Retrieved from http://www.ietf.org/rfc/rfc2712.txt

Needham, R. M., & Schroeder, M. D. (1978). Using encryption for authentication in large networks of computers. *Communications of the ACM*, *21*(12), 993–999. doi:10.1145/359657.359659

NIST. (2007). *Recommendation for Key Management*. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

Osterweil, E., Kaliski, B., & Larson, M., & McPherson. (2012). *Reducing the X*. 509 *Attack Surface with DNSSEC's DANE*. Securing and Trusting Internet Names. SATIN.

Ozment, A., & Schechter, S. E. (2006). *Bootstrapping the adoption of internet security protocols*. Paper presented at the Fifth Workshop on the Economics of Information Security. Cambridge, UK.

Payne, B. D., & Edwards, W. K. (2008). A brief introduction to usable security. *IEEE Internet Computing*, *12*(3), 13–21. doi:10.1109/MIC.2008.50

Peter, E., & Burns, J. (2010). *Defcon 18*. Retrieved December 2011, from http://www.youtube.com/watch?v=gpZ6AbkqBQo

Ramasubramanian, V., & Sirer, E. G. (2005). Perils of Transitive Trust in the Domain Name System. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. Berkeley, CA: USENIX Association.

Rescorla, E. (2008, February 27). *Comments on draft-kupwade-sip-iba-00*. Retrieved from http://www.ietf.org/mail-archive/web/sip/current/msg22283.html

Rescorla, E. (2009, July 16). *Review of draft-huang-tls-ibe-00*. Retrieved from http://www.ietf.org/mail-archive/web/tls/current/msg03611.html

Rosa, S., & Schultze, S. (2013). Trust Darknet: Control and Compromise in the Internet's Certificate Authority Model. *IEEE Internet Computing*, 18–25. doi:10.1109/MIC.2013.27

Roschke, S., Ibraimi, L., Cheng, F., & Meinel, C. (2010). Secure Communication using Identity Based Encryption. In *Proceedings of Communications and Multimedia Security* (pp. 256–267). Springer. doi:10.1007/978-3-642-13241-4_23

Schechter, S. (2013). *The User IS the Enemy, and (S)he Keeps Reaching for that Bright Shiny Power Button!* Paper presented at the Workshop on Home Usable Privacy and Security. Newcastle, UK.

Schultze, S. B. (2010). The Certificate Authority Trust Model for SSL: A Defective Foundation for Encrypted Web Traffic and a Legal Quagmire. *Intellectual Property and Technology Law Journal*, 3-8.

Shamir, A. (1985). Identity-based cryptosystems and signature schemes. In *Proceedings of Advances in cryptology* (pp. 47–53). Springer. doi:10.1007/3-540-39568-7_5

Sheng, S., Broderick, L., & Koranda, C. A. (2006). *Why Johnny still can't encrypt: evaluating the usability of email encryption software*. Retrieved from https://cups.cs.cmu.edu/soups/2006/posters/sheng-poster_abstract.pdf

Soghoian, C., & Stamm, S. (2012). Certified lies: Detecting and defeating government interception attacks against ssl (short paper). In *Proceedings of Financial Cryptography and Data Security* (pp. 250–259). Springer. doi:10.1007/978-3-642-27576-0_20

Stallings, W. (2011). Cryptography and Network Security Priniciples and Practice. *New York: Prentice Hall.*

Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., & Cranor, L. (2009). Crying wolf: An empirical study of SSL warning effectiveness. In *Proceedings of the 18th Usenix Security Symposium* (pp. 339-416). Retrieved from http://www.usenix.org/events/sec09/tech/full_papers/sunshine.pdf

Tanimoto, S., Yokoi, M., Sato, H., & Kanai, A. (2011). Quantifying Cost Structure of Campus PKI. In *Proceedings of 11th International Symposium Applications and the Internet* (SAINT) (pp. 315-320). Munich: IEEE.

Thomas, M. (2001, June). *Requirements for Kerberized Internet Negotiation of Keys* (RFC 3129). Retrieved from http://www.ietf.org/rfc/rfc3129.txt

TrustWave SpiderLabs. (2011, May). Retrieved December 2011, from http://blog.spiderlabs.com/2011/04/certificate-revocation-behavior-in-modern-browsers.html

Tsai, C.-R. (2002). Non-repudiation in practice. In Proceedings of The Second International Workshop for Asian Public Key Infrastructures. *National Taiwan University.*

Ulrich, A., Holz, R., Hauck, P., & Carle, G. (2011). Investigating the OpenPGP Web of Trust. [Berlin: Springer.]. *Proceedings of Computer Security–ESORICS*, *2011*, 489–507.

Verisign Labs. (2013, February 20). *Global DNSSEC deployment tracking*. Retrieved from http://secspider.cs.ucla.edu/growth.html

Vratonjic, N., Freudiger, J., Bindschaedler, V., & Hubaux, J.-P. (2013). The inconvenient truth about web certificates. [Springer.]. *Proceedings of Economics of Information Security and Privacy*, *III*, 79–117. doi:10.1007/978-1-4614-1981-5_5

Whitten, A., & Tygar, J. D. (1999). Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium* (pp. 169-184). Washington, DC: McGraw-Hill.

Yang, H. O. (2011). Deploying cryptography in Internet-scale systems: A case study on DNSSEC. *IEEE Transactions on Dependable and Secure Computing*, 656–669. doi:10.1109/TDSC.2010.10

Yao, D., Fazio, N., Dodis, Y., & Lysyanskaya, A. (2004). ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proceedings of the 11th ACM conference on Computer and communications security* (pp. 354-363). Washington, DC: ACM.

Ylonen, T. (1996). SSH - secure login connections over the Internet. In *Proceedings of the 6th USENIX Security Symposium* (pp. 37-42). San Jose, CA: USENIX Association.

Zimmermann, A. J. (2011, April). *ZRTP: Media Path Key Agreement for Unicast Secure RTP* (RFC 6189). Retrieved from http://tools.ietf.org/html/rfc6189

Zimmermann, P., & Callas, J. (2009). The Evolution of PGP's Web of Trust. In Beautiful Security: Leading Security Experts Explain How They Think, (pp. 107-130). Academic Press.

Zurko, M. E., & Simon, R. T. (1996). User-centered security. In *Proceedings of the 1996 workshop on New security paradigms* (pp. 27-33). Lake Arrowhead: ACM.

ADDITIONAL READING

Amann, B., Vallentin, M., Hall, S., & Sommer, R. (2012). *Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. TR-12-014*. ICSI.

Boldyreva, A., Vipul, G., & Virendra, K. (2008). Identity-based encryption with efficient revocation. In Proceedings of the 15th ACM conference on Computer and communications security (pp. 417-426). Alexandria: ACM.

Chandramouli, R., & Scott, R. (2009). Open issues in secure DNS deployment. *IEEE Security & Privacy*, *7*(5), 29–35. doi:10.1109/MSP.2009.129

Chen, L., Harrison, K., Moss, A., Soldera, D., & Smart, N. P. (2002). *Certification of public keys within an identity based system. Information Security* (pp. 322–333). Springer Berlin Heidelberg.

Garfinkel, S. L. (2003). Email-based identification and authentication: An alternative to PKI? *IEEE Security & Privacy*, *1*(6), 20–26. doi:10.1109/MSECP.2003.1253564

Gentry, C. (2003). *Certificate-based encryption and the certificate revocation problem. Advances in Cryptology - EUROCRYPT 2003* (pp. 272–293). Springer Berlin Heidelberg. doi:10.1007/3-540-39200-9_17

Gentry, C., & Silverberg, A. (2002). Hierarchical ID-Based Cryptography. [Springer Berlin Heidelberg.]. *ASIACRYPT*, *2002*, 548–566.

Herzberg, A., & Margulies, R. (2012). Training Johnny to Authenticate (Safely). *IEEE Security and Privacy*, *10*(1), 37–45. doi:10.1109/MSP.2011.129

Holz, R., Riedmaier, T., Kammenhuber, N., & Carle, G. (2012). X. 509 forensics: Detecting and localising the SSL/TLS men-in-the-middle. [Springer Berlin Heidelberg.]. *Computer Security–ESORICS*, *2012*, 217–234.

Payne, B. D., & Edwards, W. K. (2008). A brief introduction to usable security. *IEEE Internet Computing*, *12*(3), 13–21. doi:10.1109/MIC.2008.50

Schechter, S. (2013). The User IS the Enemy, and (S)he Keeps Reaching for that Bright Shiny Power Button! Workshop on Home Usable Privacy and Security. Newcastle.

Soghoian, C., & Stamm, S. (2012). *Certified lies: Detecting and defeating government interception attacks against ssl (short paper). Financial Cryptography and Data Security* (pp. 250–259). Springer Berlin Heidelberg.

Tsai, C.-R. (2002). Non-repudiation in practice. The Second International Workshop for Asian Public Key Infrastructures. National Taiwan University.

Vratonjic, N., Freudiger, J., Bindschaedler, V., & Hubaux, J.-P. (2013). The inconvenient truth about web certificates. [Springer New York.]. *Economics of Information Security and Privacy*, *III*, 79–117.

Yao, D., Fazio, N., Dodis, Y., & Lysyanskaya, A. (2004). ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In Proceedings of the 11th ACM conference on Computer and communications security (pp. 354-363). Washington: ACM

KEY TERMS AND DEFINITIONS

Asymmetric Cryptography: Traditional (*symmetric*) encryption and decryption algorithms utilize the same *secret* cryptographic key for protecting the information. Asymmetric encryption and decryption algorithms utilize pair of cryptographic keys (*key pair*): public key and private key. Public key is used for information encryption, but the encrypted information can be decrypted only by private key. This concludes that public key can be known to unauthorized actors as well, without putting the protected information at risk.

Certification Authority (CA): Certification authority is a trusted third party whose main responsibility is to issue and manage public key certificates. Certification authority should attest that identity included in an issued public key certificate was verified in a trustworthy manner. The identity verification process is typically documented in CA policies as well as other CA operational aspects. These policies are made public to relying parties that make use of the certificates issued by particular CA.

Certificate Transparency: The aim of certificate transparency is to provide publicly available audit logs that contain issued public key certificates as well as their issuing certification authority. This way the certificate holders will be able to monitor for certificates issued for them and be able to detect an impersonation attack.

Cryptographic Key: Encryption and decryption algorithms used for protecting a particular piece of information are usually known to authorized as well as unauthorized actors. In order to protect the information from unauthorized access a special parameter is used as an additional input for encryption and decryption process. This parameter is called cryptographic key.

Cryptographic Key Distribution: When there are several authorized actors involved in encryption and decryption of particular information, they need to share their cryptographic keys (either secret cryptographic key or public keys). The process of sharing these keys in a secure manner is called cryptographic key distribution.

DANE: DNS Based Authentication of Named Entities provides alternate public key distribution infrastructure by utilizing DNS and DNSSEC. DANE defines a new DNS resource record that can hold information about public key certificate association with a specified DNS domain or host.

DNSSEC: DNSSEC adds integrity and authenticity service to the DNS by utilizing asymmetric cryptography and digitally signing DNS responses. DNSSEC makes use of strictly hierarchical public key infrastructure that is coincident to the DNS hierarchy (authoritative name server acts as a certification authority for its zone).

Encryption: Is the algorithmic process of transforming information in a way that protects several its security attributes, especially one or more from following set: confidentiality, integrity and authenticity. Only authorized actors can reverse this process - *decrypt* the protected information in order to access or manipulate it.

Public Key Certificate: The certificate is a data structure that binds public cryptography key and information about its holder identity. This entire structure is public and its authenticity/integrity is protected by cryptographic means. Most widespread type of public key certificate is a X.509 certificate. X.509 certificates are used for example for protecting HTTP/SSL communications, securing email communications and verifying electronic and digital signatures.

Public Key Infrastructure (PKI): The set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke public key certificates.

ENDNOTES

1 In SSHv2 similar approach to ZRTP can be used.

2 Identity based encryption might be as well used along with hierarchical based key distribution infrastructures. See for example (Adida, Hohenberger, & Rivest, 2005).

3 These estimates come mostly from practical experience. According to (Jääskeläinen, 2013) average time to setup SSH key is 15 minutes, and approximate cost per hour of a security administrator is $60 (Jääskeläinen, 2013), (Gutmann, 2011).

The estimate for PGP key set up costs come from (Whitten & Tygar, 1999), where the users have had 90 minutes to sign/encrypt mail using PGP. From our experience it takes approximately an hour to configure PGP and train a user.

Estimates for IBE and PKI originate in a research sponsored by Voltage Security (Ferrris Research, 2006) as well as expert discussions (Gutmann, 2011), (John Callas, 2013) and our practical experience.