A Subliminal Channel in EdDSA

Information Leakage with High-Speed Signatures

Alexander Hartl
TU Wien
Institute of Telecommunications
Gusshausstrasse 25/E389
Vienna 1040, Austria
alexander.hartl@student.tuwien.ac.at

Robert Annessi*
TU Wien
Institute of Telecommunications
Gusshausstrasse 25/E389
Vienna 1040, Austria
robert.annessi@nt.tuwien.ac.at

Tanja Zseby
TU Wien
Institute of Telecommunications
Gusshausstrasse 25/E389
Vienna 1040, Austria
tanja.zseby@tuwien.ac.at

ABSTRACT

Subliminal channels in digital signatures provide a very effective method to clandestinely leak information from inside a system to a third party outside. Information can be hidden in signature parameters in a way that both network operators and legitimate receivers would not notice any suspicious traces. Subliminal channels have previously been discovered in other signatures, such as ElGamal and ECDSA. Those signatures are usually just sparsely exchanged in network protocols, e.g. during authentication, and their usability for leaking information is therefore limited. With the advent of high-speed signatures such as EdDSA, however, scenarios become feasible where numerous packets with individual signatures are transferred between communicating parties. This significantly increases the bandwidth for transmitting subliminal information. Examples are broadcast clock synchronization or signed sensor data export. A subliminal channel in signatures appended to numerous packets allows the transmission of a high amount of hidden information, suitable for large scale data exfiltration or even the operation of command and control structures.

In this paper, we show the existence of a broadband subliminal channel in the EdDSA signature scheme. We then discuss the implications of the subliminal channel in practice using three different scenarios: broadcast clock synchronization, signed sensor data export, and classic TLS. We perform several experiments to show the use of the subliminal channel and measure the actual bandwidth of the subliminal information that can be leaked. We then discuss the applicability of different countermeasures against subliminal channels from other signature schemes to EdDSA but conclude that none of the existing solutions can sufficiently protect against data exfiltration in network protocols secured by EdDSA.

*Corresponding author.

MIST'17, , October 30, 2017, Dallas, TX, USA.
© 2017 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN ISBN 978-1-4503-5177-5/17/10…$15.00
https://doi.org/10.1145/3139923.3139925

1 INTRODUCTION

Subliminal channels are a variant of covert channels that hide information by exploiting certain mathematical properties of cryptosystems. Subliminal channels can, therefore, be used to clandestinely leak information by using common cryptographic protocols. A typical use of subliminal channels in digital signatures are scenarios where encryption is not permitted or are just unusual but the use of signatures is allowed to ensure non-repudiation and integrity of messages. Examples are the classical prison scenario, where a warden monitors all communication between prisoners and typical censorship scenarios.

There exist many use cases in which integrity is of higher concern than confidentiality and therefore signatures are desirable but, at the same time, data may be sent unencrypted due to resource constraints or simply because the time needed for encryption is detrimental to the applications' requirements; examples are typical group communication scenarios such as (broadcast or multicast) clock synchronization or sensor data collection. Subliminal channels can be used in such scenarios to hide malicious data for information leakage or malware communication in an existing overt communication. In this way, subliminal channels can cause significant damages to companies or whole nations, if confidential information is leaked to the outside.

In contrast to other information hiding techniques such as obfuscation and steganography the sender of the subliminal information does not have to alter the contents of overt messages. Subliminal channels can thus also be used in scenarios where the signer has limited or no influence to the transmitted message or modifying contents would raise suspicion.

In classical security protocols, such as IPsec or Transport Layer Security (TLS), digital signatures are usually only used sparsely, mainly in the authentication phase, which bounds the bandwidth for data leakage to a few bytes per connection. A subliminal channel in a signature scheme in a security protocol may still be sufficient to leak critical information such as keying material or status information despite limited bandwidth. The advent of high-speed signatures such as EdDSA [1] does change this bandwidth limitation though. EdDSA was primarily designed for small signature size as well as high-performance, i.e., fast signing and verification. With these properties it can be used in scenarios where many subsequent packets need to be signed, like clock synchronization information or sensor data collection. This results in a significant increase in the bandwidth usable for subliminal information.

In this paper, we show the existence of a subliminal channel in EdDSA and describe how the channel can be used in different scenarios. We analyze the bandwidth that can be achieved and evaluate potential countermeasures. We distinguish two cases: (1) the legitimate sender wants to transmit subliminal information intentionally, and (2) the legitimate sender has been compromised and the subliminal message is inserted by malware, which wants to remain undetected and has access to the signing process. Then, we elaborate in which network protocols EdDSA is in use or may be used in the near future in order to assess in which way subliminal channels may be employed. We furthermore identify how much (subliminal) information can be clandestinely transferred via these network protocols by exploiting the subliminal channel in EdDSA. Finally, we evaluate potential countermeasures that prevent, mitigate, or at least detect the use of subliminal channels.

Our main findings are:

(1) A broadband subliminal channel in EdDSA does exist and can be exploited by sharing the signing key with the receiver of the subliminal information. Such prior key sharing is a common requirement in many subliminal channels and can be conducted by sharing the key a priori through external communication, using a narrowband subliminal channel, or by compromising the sender.
(2) A narrowband subliminal channel in EdDSA can be established, which does not require knowledge of the signing key by the subliminal receiver. The narrowband channel can be used to send the required signing key to the subliminal receiver in order to establish the broadband subliminal channel.
(3) The maximum bandwidth of the broadband subliminal channel is 252 bit per signature, which is substantial given the fact that EdDSA's signature size is only 512 bit.
(4) Our analysis shows the applicability of the subliminal channel in practical experiments for very different scenarios, such as broadcast clock synchronization, signed sensor data collection, and the TLS handshake. The maximum achievable bandwidth is investigated in practical experiments.
(5) We identify three countermeasures that ensure subliminal-freeness: (1) an extension based on pre-published nonce points, (2) a scheme based on active warden interaction, (3) the use of zero-knowledge proofs. Additionally, we describe a method to detect subliminal channels based on checking whether signatures of two identical messages are themselves identical. Our analysis shows that none of the countermeasures is generally viable in the context of network protocols so that protecting information assets from leakage remains a major challenge. Also, the detection methods may be circumvented by a skilled adversary.

We conclude that when employing EdDSA in network security protocols, network operators, security engineers, as well as protocol designers should be aware of the existence of the subliminal channels described in this paper. If the possibility of having a subliminal channel cannot be tolerated in a particular application area, either a different, subliminal-free signature scheme (with potentially less attractive properties than EdDSA) should be employed instead or significant resources are required to ensure that a subliminal channel is not actively exploited.

2 RELATED WORK

The concept of subliminal channels was first introduced in 1984 by Simmons [2]. Simmons imagined two prisoners who are allowed to communicate with each other in terms of messages. As the prison warden aims to prevent the prisoners from coordinating an escape plan, he only passes on messages that are unencrypted so that he can read them. On the other hand, the prisoners fear of the warden forging messages from the respective other such that they insist on the communication being authenticated using signatures. With this setting in mind, Simmons shows that information can be embedded in the signature such that it does not hamper successful verification of the signature.

Simmons showed how to construct narrowband channels that allow transmitting only a few bit as well as broadband channels that allow a significant amount of subliminal information to be added to a signature [3]. These broadband channels often require the receiver of the hidden information to know the signer's secret key in order to recover the subliminal information. A noteworthy exception is the Newton channel, which was shown by Anderson et al. for the ElGamal signature scheme specifically [4]. Here, the signer unveils as many bits of information of the secret key to the subliminal receiver as should be available for the subliminal channel afterwards. So far, subliminal channels have been shown to exist in many traditional signature schemes such as DSA [3, 5], ECDSA [6, 7, 8] or RSA [9, 10, 11], and finding a mode of operation that is provably subliminal-free often turns out to be a difficult task.

In this paper we focus on the possibilities to transmit subliminal information in the emerging EdDSA signature scheme. As EdDSA experiences increasing deployment, we show how a broadband subliminal channel can be created and discuss scenarios in which these channels are used to leak information. Finally, we discuss approaches that prevent the randomness, which is essential to the signature scheme, from being (mis)used. Since all potential countermeasures come at a cost and none of them is generally applicable, our evaluation also provides valuable information for protocol designers in how to cope with the subliminal channel.

3 THE EDDSA SIGNATURE SCHEME

EdDSA [1] was introduced in 2011 by Bernstein et al. as a well performing alternative to today's signature schemes in terms of speed and security. It uses point addition on the twisted Edwards curve

$$E = \left\{(x, y) \in \boldsymbol{F}_p \times \boldsymbol{F}_p : -x^2 + y^2 = 1 + dx^2y^2\right\}, \qquad (1)$$

where $\boldsymbol{F}_p$ denotes the Galois field of order p. The scheme has several parameters: the prime p, the parameter d defining the curve, a base point $B \in E$, the order of B denoted as L, and a cofactor 2^c with integer c such that $2^c L = |E|$ (the number of points on the curve). Furthermore, a hash function H is used that produces $2b$ bit output, where $b \in \mathbb{N}$ determines the security level provided. For Ed25519, which is EdDSA used together with the curve Curve25519 from [12], these parameters are standardized in [13]. The RFC defines a further scheme, Ed448, which uses an untwisted Edwards curve [14] and provides 224-bit security rather than 128-bit.

The secret key k should have an entropy of at least b bit. It is mapped to a $2b$-bit string $h = H(k)$. Bits h_c to h_{n-1} with $c \leq n < b$

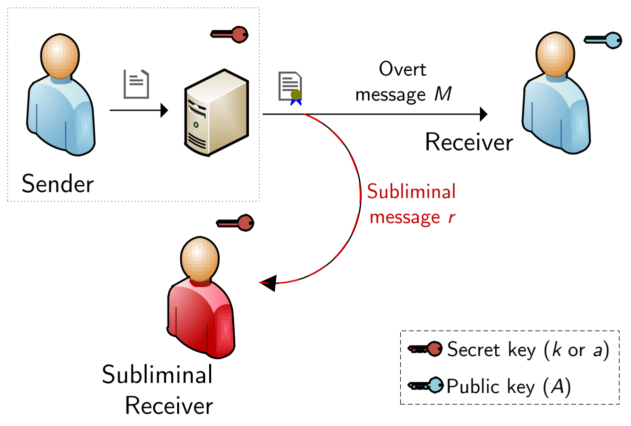

Figure 1: A subliminal channel in EdDSA.

of h are in turn injectively mapped to a number a. Knowledge of a is sufficient for producing valid signatures, which justifies considering a as the signing key. The public key consists of a point on the curve $A = aB$.

To generate a signature for a message M, first a nonce value

$$r = H(h_b, \ldots, h_{2b-1}, M) \tag{2}$$

has to be derived. The signature consists of two parts: (1) a point $R = rB$ and (2) a number $S = (r + H(R, A, M)a) \bmod L$. For verification the receiver has to check the group equation

$$2^c SB = 2^c R + 2^c H(R, A, M)A. \tag{3}$$

EdDSA is based on a digital signature scheme that was first described by Schnorr [15]. A main concern when using this kind of signatures is that r has to be chosen unpredictably [1]. Indeed, if r can be guessed correctly for an existing signature, then the signing key a can be simply computed as $a = (S - r)/H(R, A, M) \bmod L$ using the extended Euclidean algorithm [1]. Furthermore, if the same nonce value has been used for generating signatures of different messages M_1 and M_2, the signing key a can be found as well as $a = (S_1 - S_2)/(H(R, A, M_1) - H(R, A, M_2)) \bmod L$. Both issues can be addressed by deriving r from the message and the secret key as done for EdDSA (see Eq. (2)). This is in contrast to ECDSA, where the issue of choosing an appropriate value for r is left to the implementation, which has to use a pseudorandom number generator for this purpose.

4 SUBLIMINAL CHANNELS IN EDDSA

In [3] Simmons introduced a classification of subliminal channels according to the bandwidth of subliminal information. Thus, in the case of a *broadband* subliminal channel the information can use almost all the signature's bits that are not needed for its security against forgery. In the case of a *narrowband* subliminal channel the subliminal bandwidth is significantly smaller and often as small as just a few bits. EdDSA yields a broadband subliminal channel as well as a narrowband channel.

4.1 The Broadband Channel

Like in other signature schemes that are based on the discrete logarithm problem the (random) nonce r can be calculated with little effort from a valid signature if the signing key a is known as

$$r = S - H(R, A, M)a \bmod L \tag{4}$$

It is noteworthy that the calculation rule for r in Eq. (2) only serves as a high-quality random number generator for the signature scheme. Using a different value for the nonce does not harm the successful verifiability of the produced signature in any way. Hence, the value of r can be used as a subliminal channel by encoding covert data into it on the sender side. This data can then be recovered using Eq. (4) by anyone who holds the signing key a and is able to intercept the message and its signature (see Fig. 1). Since information can only be encoded in the residue class modulo L, the subliminal channel has a theoretical bandwidth of $\log_2 L$ bits per signature. For Ed25519 this corresponds to a bandwidth of 252 bit per signature. For Ed448 this corresponds to a bandwidth of 447 bit per signature.

The general usage of establishing a subliminal channel is depicted in Fig. 1. The sender cooperates with a subliminal receiver, who can be co-located with the receiver or reside somewhere on the network path. It is required that the receiver of the subliminal information also knows the signing key a. For this, we distinguish two cases: (1) the legitimate sender wants to transmit subliminal information intentionally and (2) the legitimate sender has been compromised and the subliminal message is inserted by malware that has access to the signing process.

For case (1), the sender directly shares the signing key with the subliminal receiver before the subliminal communication starts. By knowing the signing key, the subliminal receiver would be also capable of forging arbitrary signatures on behalf of the sender. Nevertheless, for subliminal communication scenarios we assume that the sender of the subliminal information and the subliminal receiver collaborate and that it is therefore reasonable to assume that they share the secret key k (to derive the signing key) or the signing key a upfront. For case (2), the adversary needs to clandestinely leak the signing key to the receiver. For the attack scenarios described in Section 5 the key could be leaked by using the narrowband channel described below.

Note that if data is processed with the encrypt-then-sign method the message M is the ciphertext and not the plaintext, which makes the subliminal channel usable even if the original plaintext message is encrypted and therefore unknown to the subliminal receiver (see Section 5).

4.2 A Narrowband Channel

Besides the broadband subliminal channel described above, it is also possible to use the signature as a narrowband subliminal channel. In this case the sender tries to make the encoded representation of the nonce point R show a specific bit pattern like, for example, the last byte being equal to the intended subliminal information. Since computing logarithms in finite fields is infeasible, he is not able to directly choose R appropriately. However, trying many randomly picked values for r, the sender is eventually able to find a value, for which R has the desired properties. If he uses this approach the sender has to test 2^{B_s} values for r on average, where B_s denotes the desired bandwidth of the subliminal channel in bits. Due to the exponential growth with bandwidth, it is not possible to use a significant portion of the signature's bits for the subliminal information, which explains the classification as narrowband channel. This subliminal channel represents a very general approach, that can be used for many signature schemes, that either explicitly consume randomness for signature generation or implicitly allow many valid signatures for the same message. The major advantage for the attacker compared to the broadband channel described above, is that the subliminal receiver does not need to know the signing key a.

5 ATTACK SCENARIOS

In general, digital signature schemes are used in a multitude of different scenarios, which range from signing documents or emails to the use of signatures for providing authentication of communication partners in complex security protocols. Due to its excellent properties regarding performance and security, EdDSA has been proposed for a wide variety of applications, protocols, and use cases in the past[1].

In addition to these existing use cases, EdDSA was proposed in scenarios where a sender transmits many packets and needs to ensure integrity and data origin authentication to the receiver(s) [16, 17, 18]. EdDSA is an excellent choice for such scenarios as it is fast and lightweight enough to be applicable to scenarios with high sending rates and limited resources, such as broadcast clock synchronization and sensor data collection in the smart grid.

The subliminal receiver needs knowledge of the signature and the signed message to recover the subliminal information in r. If data is sent unencrypted, like in the classical prison scenario mentioned in Section 2, the exploitation of the subliminal channel is easy because any eavesdropper has access to the message. The subliminal channel is also usable if data is encrypted with the encrypt-then-sign method, which in our considered scenarios is the preferred method, for example using a scheme as described in [19]. The data is first encrypted and the ciphertext is signed. In this case the ciphertext is the message input to the signature algorithm and the subliminal receiver only needs the ciphertext (and not the plaintext message) and the signature to recover r. Both are visible to an eavesdropper in the network. Nevertheless, one case where encryption causes difficulties is if the message is signed and then encryped. In this case the subliminal receiver on the path needs to know the decryption keys as well to recover r. An example is the encrypted exchange of signatures in TLS 1.3 (see Section 5.3).

For this paper we selected three different use cases. In two use cases the authenticity of the data is of higher concern than its confidentiality so that data may be sent unencrypted (or would use the encypt-then-sign method). These use cases directly correspond to the classical prison scenario in the original work by Simmons 1984. In addition, we show a third use case, the use of subliminal channels in TLS where in some cases the signature is part of the encrypted information.

As exemplary use cases we investigate the following scenarios:

- Broadcast clock synchronization: Clock synchronization protocols, such as the Network Time Protocol (NTP) and the Precision Time Protocol (PTP) can use high-speed signatures for signing messages to provide data origin authentication. Time synchronization protocols require data origin authentication to ensure that the time information comes from a trusted server and has not been modified on the path. Confidentiality of time information is of less concern, so data may be sent unencrypted. Time synchronization protocols are used in many environments and therefore provide a broad infrastructure for transmitting subliminal information. We propose to use high-speed signatures in such scenarios [16] and as an example show the use of EdDSA in NTP broadcast mode.
- Smart grid sensor data collection: High-speed signatures can be used to protect data sent from sensors to data collectors in smart grid monitoring. Depending on the amount of sensor data, many signatures need to be sent in short time intervals. Again, data origin authentication is of higher concern than confidentiality since modified sensor data may lead to wrong control decisions. Also for such scenarios we propose the use of high-speed signatures. As an example, we show the use of EdDSA for transmitting phasor measurement data in a smart grid environment, transmitting 60 to 120 packets per second and therefore providing a high bandwidth for subliminal information.
- Network security protocols, such as IPsec and TLS use signatures to provide authenticity of the communicating partners. Usually only few signatures are exchanged. But even those infrequent transmissions can be used to leak critical information such as keying material. As example, we show the use of EdDSA in TLS 1.2 and prior versions [20, 21] and TLS 1.3 [22].

In the following we show and implement attack scenarios using the subliminal channel for the above use cases.

5.1 Broadcast Clock Synchronization

A scenario where the need for signing numerous messages arises naturally is broadcasting time information using clock synchronization protocols such as NTP and PTP. An authentication scheme for this purpose has to satisfy various requirements to minimize the probability of an attacker compromising the system times of network members and thereby also compromising the proper functioning of security protocols that rely on accurate time information [16].

EdDSA seems particularly well-suited for this purpose as it provides adequate performance in terms of signing and verification speed and achieves good security properties. However, the possibility of embedding subliminal information in the signatures has to be taken into account in security-sensitive environments. A signature scheme is an attractive candidate for carrying subliminal information because it may yield a large bandwidth, due to the large number of packets. Furthermore, clock synchronization protocols are widely deployed throughout the Internet, leading to a broad infrastructure usable for leaking information through subliminal channels. Additionally, when time synchronization messages are broadcast over the network, the need for the subliminal receiver to

[1]For projects that employ EdDSA see https://ianix.com/pub/ed25519-deployment.html.

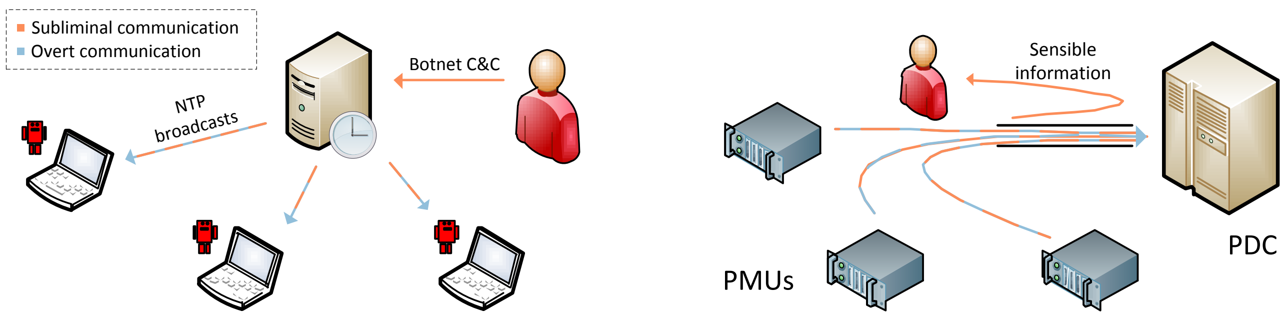

Figure 2: Attack scenarios for the subliminal channel in EdDSA: Botnet Command and Control using NTP broadcasts (left) and information leakage for phasor measurements (right).

take special measures for eavesdropping on the signed packets is eliminated as any network access suffices.

Exemplary use cases for such a subliminal channel are the clandestine leakage of information through a company's network or the operation of a botnet where the signature of the NTP packets are exploited to transmit command and control messages to bots implemented in NTP clients. Even if the interval of time broadcasts is large the bandwidth may suffice for these purposes.

5.2 Smart Grid Sensor Data

There are several reasons why components that operate in smart grids and other cyber-physical system environments are deemed highly security critical. Availability, integrity, and data origin authentication are essential for sensor data collection in smart grids. Confidentiality is usually of less concern and encryption often omitted due to time and resource constraints. In such a security-critical setting a signature scheme is of fundamental importance.

Since some applications have real-time requirements and some devices are built on low-power hardware, lightweight signature schemes with fast signature generation and verification are required. For these reasons, we propose that high-speed signatures, such as EdDSA, should be used as signing method for transmitting sensor data in future deployments. Nevertheless, when deploying such signatures the existence of subliminal channels needs to be taken into account.

An example of a possible subliminal channel in a smart grid scenario appears in the course of transmission of data measured by Phasor Measurement Units (PMUs). Here, the need for a fast signature scheme arises from the fact that 60 to 120 measurement values per second are transmitted to data concentrators and may need to be processed in real-time (such as in control applications for example). Being able to place information in the signatures of this large number of signed packets implies a large bandwidth of subliminal information. Exploiting this possibility, a substation could for instance embed subliminal information into signatures in order to exfiltrate critical information such as its location, configuration parameters, maintenance schedules, or even key material. In this case, subliminal data transmission is restricted to the network of sensors and data collectors. However, since sensor deployment in smart grids and other environments is growing continuously, the infrastructure usable for subliminal channels grows as well.

5.3 Network Security Protocols

TLS is the prevalent protocol for securing communications on today's Internet. It is widely deployed and therefore an attractive carrier for information leakage and other communication hiding use cases. Hence, it seems remarkable that only a small amount of research has been conducted into exploring ways to covertly transmit information using the TLS protocol so far.

Some work has been done to identify covert channels in TLS, which use protocol fields in the header or packet timing to exchange covert information. Most publications propose to use the randomness field that both client and server transmit in their *Hello* messages [23, 24, 25, 26, 27]. Other options are to use the session id field [23], encryption padding [23], timestamp values in the *Hello* messages [23], the certificates [25] or the exchanged cipher suites lists [23, 25]. Subliminal channels, on the other hand, hide information in the cryptographic protocol. Compared to covert channels, significantly fewer subliminal channels have been identified.

When hiding information in TLS, it is relevant at which stage of the connection establishment the subliminal information is injected, because parts of the TLS connection establishment are already encrypted. This includes the signature exchange in TLS 1.3 (Fig. 3 on the following page). In such cases, the subliminal receiver does not only need the signing key to recover the subliminal information but also requires the shared secret key of the communication partners in order to decrypt the ciphertext that contains the signature.

5.3.1 *RSA ciphertext.*

Gołębiewski, Kutyłowski, and Zagórski [26] describe a method for exploiting the TLS premaster secret as a hidden information channel if RSA is used for the key exchange. If RSA is used, the premaster secret is chosen by the client, encrypted under the server's public key and sent to the server in the *Client Key Exchange* message. As neither subliminal sender nor subliminal receiver hold the server's secret key, the value cannot be used as a broadband channel: The subliminal receiver is not able to decrypt the ciphertext. Thus, the subliminal information must be contained in the ciphertext instead of the premaster secret itself. On the other hand, lacking the server's secret key also the subliminal sender is not able to choose the premaster secret in a way that makes the ciphertext equal to the intended subliminal information. Hence, similarly to the narrowband subliminal channel from section 4.2 he has to repeatedly try different values for the premaster secret until he finds a value that makes the encrypted value show a pattern that corresponds to the subliminal information.

5.3.2 Diffie-Hellman parameters. One possibility similar to the RSA approach described by Gołębiewski, Kutyłowski, and Zagórski [26] is to use the exchange of parameters in the Diffie-Hellman key exchange. Using Diffie-Hellman is one option for key negotiation and in the course of this process signatures are used for authentication purposes. A narrowband subliminal channel can be established in a manner similar to that described in Section 4.2: Trying different values for α, a value of g^α can be found that shows the desired bit pattern.

However, we are not aware of any possibility for using this method as a broadband subliminal channel, as it is not feasible to directly encode subliminal information in the transmitted value g^α. The discrete logarithm α has to be known by the sender in order to calculate the joint secret key. If the sender wants to transmit information directly in g^α he would need to find an α that generates the subliminal message g^α. Thus, since computing the discrete logarithm is infeasible, these points g^α cannot be chosen equal to the intended subliminal information.

5.3.3 Signatures. In contrast to the two subliminal channels described above, digital signatures can provide the opportunity to establish broadband subliminal channels in TLS. During the handshake signatures are used to prove the identity of the server and, optionally, of the client. Signatures furthermore ensure the integrity of the Diffie-Hellman key exchange and are an essential building block for the certificate chain used to verify the server's identity based on a set of certificate authorities.

If ephemeral Diffie-Hellman is used for key exchange, the authenticity of the server is verified using a signature. Fig. 3 shows the messages that are exchanged during a handshake. In protocol versions prior to TLS 1.3, the signature is sent together with the server's Diffie-Hellman parameters in the *Server Key Exchange* message. Besides these parameters the signed data only contains the random values from the client's and the server's *Hello* messages. At that point the data exchange is unencrypted and the signed message (which is needed to recover subliminal information in the signature) is therefore known by anyone eavesdropping on the connection. Furthermore, as described in Section 7.2, the inclusion of these random messages hampers detection of the subliminal channel because signatures for the same signed data will hardly ever occur. The client can also be authenticated using a certificate. In this case the corresponding signature is transmitted in the *Certificate Verify* message and is computed over all handshake messages up to the current point. When using client authentication, subliminal channels exist in both directions, otherwise only the subliminal channel from the server to the client can be exploited.

In contrast to earlier TLS versions, the use of ephemeral Diffie-Hellman is enforced in the upcoming version TLS 1.3 with the associated parameters being exchanged already in the client's and the server's *Hello* messages. The signatures used for authentication are exchanged in the *Certificate Verify* messages. The signed data now contains the entire handshake up to the respective *Certificate Verify* message for both server and client. The most important difference in TLS 1.3 compared to earlier versions is the fact that the handshake data is now encrypted as soon as the shared secret from

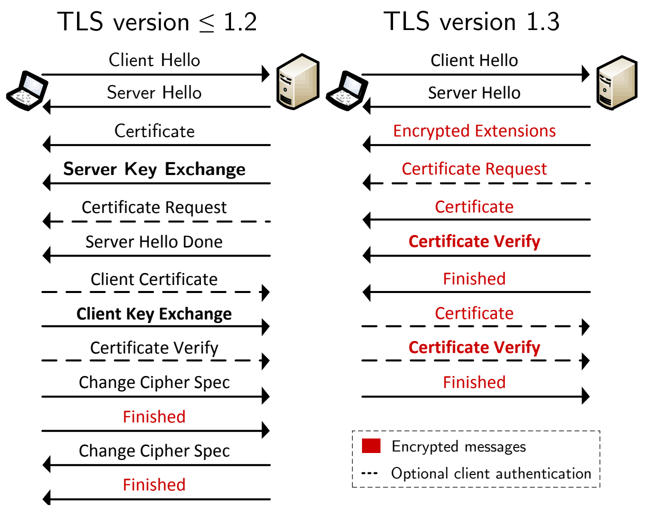

Figure 3: The TLS handshake.

the key exchange algorithm is available. Therefore, all messages following the *Server Hello* are unavailable to a passive eavesdropper who does not know the encryption key. For this reason, an eavesdropper cannot see the signed message and therefore cannot recover the subliminal information in r. So in order to use this as subliminal channel the subliminal receiver additional needs the encryption key.

Finally, the signatures used for building the certificate chain can equally be used to carry hidden data in the course of the TLS handshake if the subliminal sender is the issuer of one of the certificates contained in the certificate chain. But this approach may be easy to detect in comparison to above mentioned signatures.

Table 1 on the next page provides references for subliminal channels discovered that may be used in TLS. Current TLS versions support either DSA or ECDSA when using the Diffie-Hellman key exchange. EdDSA is one of the new signature scheme options available in TLS 1.3 [22] and is proposed to be used with version 1.2 and earlier TLS versions in [20]. Therefore, it is likely that EdDSA is implemented in current and future implementations of TLS and the subliminal channel in EdDSA can be utilized to transmit hidden information.

5.4 Further Attack Scenarios

In addition to the attack scenarios already investigated in this section, further scenarios for the use of the subliminal channel in EdDSA are likely and should be taken into account when applying EdDSA.

Subliminal channels can be used to encode additional information in digital signatures in passports [6] or health insurance cards. The issuer could include subliminal information that provides additional information without the owner's knowledge. Also, DNSCrypt[2] uses Ed25519 and DNSSEC [28] supports it, which is why these protocols are susceptible to subliminal channels. Furthermore, upcoming efforts to secure Internet Inter-AS routing in the Border

[2]https://dnscrypt.org/

Table 1: Subliminal channels in TLS.

Exploited scheme	Protocol version	Bandwidth	Difficulty of detection[1]	References
Key exchange using RSA	$\leq$ TLS 1.2	narrow/broad[3]	hard	[26]
Diffie-Hellman key exchange	all	narrow	hard	
DSA	$\leq$ TLS 1.2	narrow/broad[3]	hard	[3]
ECDSA	all[2]	narrow/broad[3]	hard	[6]
EdDSA	all[2]	narrow/broad[3]	hard	this paper

[1] Difficulty of detection assumes subliminal information that is indistinguishable from uniformly random data.
[2] For TLS 1.3 only if the shared secret for encryption can be leaked to the receiver of the covert data.
[3] Depending on the receiver of the covert data possessing the corresponding secret key.

Gateway Protocol (BGP) use nested signatures for path validation. In these signatures, subliminal information could be transmitted between BGP routers. Also, many cryptocurrencies employ EdDSA for signing transactions, which might be exploited to add subliminal information. These are just a few examples for the applicability of the subliminal channel in EdDSA.

6 MEASUREMENT RESULTS

We performed experiments for the three attack scenarios described in the previous section in order to show the possibility to transmit subliminal information within the EdDSA signature in practice and to measure the actual achievable bandwidth in the different scenarios. In this section we present both the experimental setup and the measurement results.

6.1 Clock Synchronization: Signed NTP

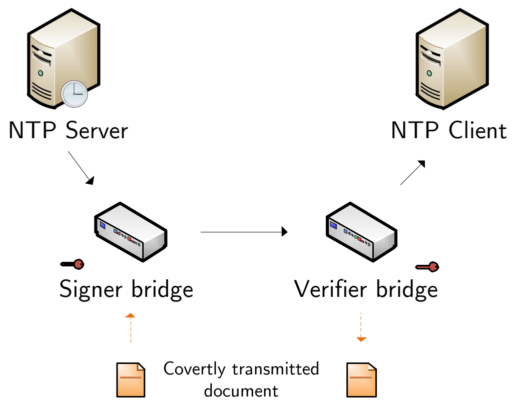

Figure 4: Experimental setup for investigating a subliminal channel in signed broadcast NTP messages.

The following experimental setup was used (Fig. 4): the NTP server and client run unmodified NTP software while the signature generation is being performed by the network bridges, which are located between the NTP server and the NTP client. The subliminal information is therefore embedded by the 'signer bridge'. The subliminal receiver can thus be anywhere on the broadcast domain. In our case, the receiving part is performed by the 'verifier bridge'. Hence, unlike the normal signing situation, here the verifier also has to hold the signing key that was used to generate the signature. As described in Section 4, the sharing of the signer's key is required often for using broadband subliminal channels in signatures.

For the setup we employed machines running Debian Linux 'Jessie' as operating system and used iptables in conjunction with nfqueue on the bridge devices to sign packets and remove signatures, respectively. We used the cryptographic primitives from the NaCl[3] library to perform the tasks of signature generation and verification. In order to implement the retrieval of the subliminal information according to Eq. 4 on page 3, we added a function for performing subtractions in the appropriate finite field. Apart from this modification, the nonce value was substituted in the signing process to include the subliminal information and was recovered from the signature. In this way, the subliminal channel was proven operational.

In such practical setting, it is more difficult to handle partial bytes. For this reason, we transmitted 248 bit or 31 B of subliminal information per message, instead of the theoretically possible 252 bit. Due to the broadcast interval of 8 seconds, a bandwidth of approximately 3.9 B/s was achieved. 8 seconds is the smallest broadcast interval possible in NTP and, hence, 3.9 B/s is the largest achievable bandwidth that is possible without modifying NTP's source code.

6.2 Smart Grid Communication: PMU Sensor Data Transmission

For the second experiment, we reused the signer and verifier bridges from the previous setup (Fig. 4) to investigate the possibility of hiding information in signed PMU measurements. Instead of the NTP server, however, the measurement device employed was a phasor measurement unit 1133A Power Sentinel by Arbiter Systems, which broadcasts measurement data or sends it to a specific receiver (such as a phasor measurement data concentrator). Such devices are used to measure the phasors of electric current and voltage at different locations in smart power grids and can send up to 120 packets per second.

In our scenario we used a standard configuration sending 10 packets per second and added an EdDSA signature to each packet. We

[3] https://nacl.cr.yp.to/

used the manufacturer's proprietary PowerSentinelCSV protocol, which transmits 10 UDP packets of measurement data per second achieving 310 B/s of subliminal bandwidth. In principle the same bandwidth could also be achieved using other protocols such as IEEE C37.118, defined for PMU data transmission, if available on the devices.

6.3 Security Protocol: Key Recovery for TLS 1.2

We examined the EdDSA subliminal channel in TLS from Section 5.3 by using a setup consisting of an nginx 1.13.0[4] webserver and a simple HTTP client application, both compiled with Google's OpenSSL fork BoringSSL[5] which supports Ed25519 as well as the current TLS 1.3 draft. Similarly to the extension of NaCl in the two previous experiments, we extended the functionality of BoringSSL by a routine to perform finite field subtractions. Furthermore, minor code modifications were conducted to allow the nginx webserver to use TLS 1.3.

Given these modifications, Eq. (4) was implemented in order to recover the subliminal information, which worked as expected for both TLS 1.2 and TLS 1.3 with a bandwidth of 31 B per handshake[6]. Table 2 provides a summary of the measurement results for all three scenarios. Especially with the transmission of sensor data a high amount of subliminal data can be included.

Considering TLS 1.2, an example of information that may be transmitted using this subliminal channel is keying material, which allows a passive eavesdropper to decrypt the data meant to be protected by TLS. Another scenario is leaking information. In this case, the method yields advantages to an adversary as, if done properly, no anomaly detection technique will detect the information flow. Furthermore, it is noteworthy that, in normal protocol operation the signed data will never be the same for two handshakes, since random values from both server and client are included, which further reduces the risk of detection.

Table 2: Measurement results for all three scenarios.

Scenario	**Bandwidth**
NTP broadcasts	max. 3.9 B/s
PMU measurements	310 B/s
TLS handshake	bidirectionally 31 B

7 APPROACHES TO MITIGATE SUBLIMINAL COMMUNICATION

Some approaches that aim at mitigating or at least detecting subliminal communication have been published for other signature schemes. In this section, we investigate whether any of these methods can be applied to protect EdDSA signatures against the establishment of subliminal channels.

[4]http://nginx.org/
[5]https://boringssl.googlesource.com/boringssl/
[6]In a practical scenario it has to be taken into account that session resumption is used if many subsequent connections occur. In this case, the achievable bandwidth would be lower.

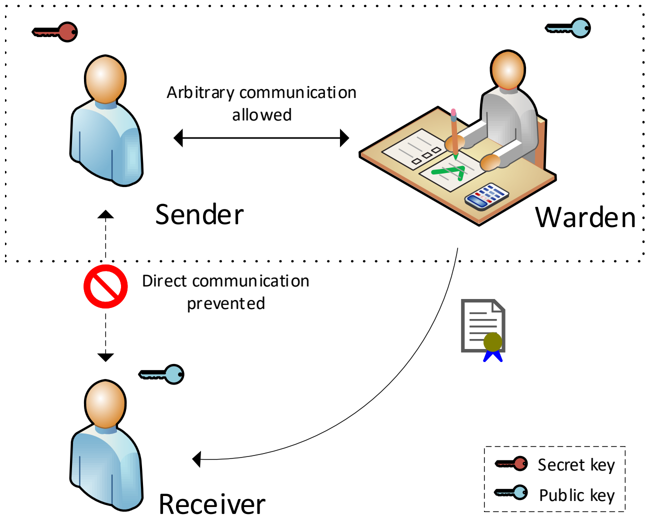
Arbitrary communication allowed
Sender
Warden
Direct communication prevented
Receiver
Secret key
Public key

Figure 5: Warden scenario to achieve subliminal-freeness.

In order to elaborate on mitigation possibilities, we introduce a 'warden' to the subliminal channel scenario established in Section 4 (see Fig. 5). The warden is a trusted instance that is located between signer and receiver[7] and tries to detect or prevent the transmission of subliminal information. This directly relates to the scenario in the original publication by Simmons [2] introduced in Section 2, where prisoners can only communicate in an unencrypted but authenticated manner. All communication is supervised by a prison employee, the warden, who may passively monitor or actively modify the communication. When using a subliminal channel, the prisoner needs to ensure that the warden is not able to detect the misuse of the signature. Since the subliminal channel in EdDSA proposed in this paper is employed in the random number used to generate the signature, the verification process in general would not show any irregularities. Nevertheless, under certain conditions (described below) the existence of a subliminal channel may be suspected though.

7.1 Ensuring Subliminal-Free Signatures

In some situations it is possible to ensure that no subliminal information is embedded in a signature. This is possible for signature schemes that allow only a single valid signature for a given message. As pointed out by Bohli, Vasco, and Steinwandt, however, it is also possible to achieve subliminal-freeness, if the signer can prove to the warden that the signature has been created in a way that only permits one valid signature [6]. In this case, the warden must have the power to discard invalid messages.

7.1.1 Pre-published Nonce Points. Using a setting where the warden can discard (invalid) messages, the simplest solution is to require the signer to generate and publish a list of R-values before the signer knows the information that is to be transmitted clandestinely during the signing process. During signature generation, the signer has to use the values in the same order as they appear in

[7]The logical instance of a warden may also be co-located with the receiver.

the list. With the nonce r being fixed a priori the signature becomes indeed unique in the sense that for a given message just one possible value for S remains. A variant of this approach would be to index the list by a number derived deterministically from the message. However, this causes an increasing number of messages to be unsignable as the corresponing signature would require a nonce that was already used earlier and would therefore make the signature reveal the signing key.

This approach has several disadvantages. First of all, due to the limited number of usable R-values, the number of distinct messages that can be signed is equally limited. When using the indexed variant, this is even more serious, as the number of unsignable messages would become significantly large once a certain amount of values has been used. Secondly, the warden needs to store the list of R-values, which may lead to significant storage requirements. For each message the signer may sign, 32 B storage are required. The most important drawback is the fact that also the transmission of the list of R-values provides a way for embedding subliminal information. Therefore, a subliminal channel is just shifted to an earlier time instant.

7.1.2 Warden Interaction. Zhang et al. proposed an interactive scheme (based on Schnorr signatures) for subliminal-free signing, in which the warden actively contributes to signature generation [29]. To prevent a subliminal channel, a total of six messages has to be exchanged between signer and warden for each signature. The scheme is shown to be secure against existential forgery if the computational Diffie-Hellman assumption holds. Furthermore, embedding subliminal information in the signature is shown to be as hard as computing discrete logarithms on behalf of the signer.

Since the EdDSA scheme is derived from Schnorr signatures, the mitigation strategy is applicable also to EdDSA. Nevertheless, the major drawbacks of the scheme are the large number of messages to be exchanged and the computational effort required both at the signer and at the warden in order to generate signatures. These drawbacks conflict substantially with the requirements in typical scenarios where EdDSA can be employed: low computational effort, fast signing and verification, and small overhead (and sometimes even only unidirectional communication).

7.1.3 Zero-knowledge Proofs. Bohli, Vasco, and Steinwandt described a provably subliminal-free signature scheme that does not require active participation of the warden [6]. Instead, the random number r required by the signature scheme is generated deterministically from the message and a proof is given to the warden that the value has indeed been derived correctly without providing means to the warden for deriving that value on behalf of the sender. This is accomplished using Naor and Reingold's pseudo random function [30]:

$$f_{p,q,\boldsymbol{a}}(\boldsymbol{x}) = g^{a_0 \prod_{1 \le i \le m, x_i = 1} a_i}. \tag{5}$$

Here, p and q are prime numbers with $q|p-1$ and $g \in \mathbb{Z}_p^*$ is the generator of a cyclic group of order q. $\boldsymbol{x} \in \{0,1\}^m$ is the seed for the random number, which in our case is a hash of the message with appropriate length $m \in \mathbb{N}$. $\boldsymbol{a} \in \mathbb{Z}_q^m$ is chosen at random by the signer during key generation and can be regarded as a secret key for signature generation. These values are chosen once for all signatures, and all values are transmitted to the warden, except for $\boldsymbol{a}$, for which only commitments are transmitted. Using this construction zero-knowledge proofs are derived, which prove that the signature has been computed deterministically from the message's hash. The proof itself is not guaranteed to be subliminal-free and, hence, must be stripped off by the warden after verification.

This provably subliminal-free signature scheme is formulated for ECDSA. Since it solves the general problem of showing that a curve point has been generated according to some specific method from the message without disclosing the point's discrete logarithm, however, it can be equally applied to EdDSA. Compared to the previous interactive approach, the approach has the advantage of simplifying the communication pattern between signer and warden. Bohli, Vasco, and Steinwandt proposed to use the scheme for passports where it should be possible for the passport's holder to make sure that the issuing party has not embedded information in the signature. Since one proof (for a security level of 128 bit) takes several megabytes, the bandwidth requirements between signer and warden are high though – too high in the context of network protocols.

Alternatively, if the requirements for the warden are relaxed to being able to prove the existence of a subliminal channel when examining a random sample of signed messages, the scheme can also be used for a scenario where it is not feasible to place a warden in a man-in-the-middle position: The signer can be obliged to offer the proofs for the generated signatures on a protected interface. A signature that has been intercepted unnoticeably can then be tested for having been generated validly. In this scenario the signer does not have to compute and store the proofs for all signatures, which would cause very high storage requirements. Instead, when requesting a proof the warden can provide the signer with the message in question, which suffices to reproduce the signature and generate the corresponding proof. In this case, however, the signer must make sure that the warden already has a valid signature for the message, as he would otherwise sign arbitrary messages on behalf of the warden.

Table 3 summarizes the three distinct approaches that mitigate subliminal communication with respect to their advantages and drawbacks.

7.2 Detecting Subliminal Communication

Although EdDSA cannot be made subliminal-free (without introducing serious disadvantages), there may still exist ways to detect the subliminal data transfer or at least to check whether subliminal data exchange can be suspected. In this section, we highlight situations in which the subliminal channel can lead to observable suspicious patterns that help to detect the transmission of subliminal information.

7.2.1 Identical Messages. Due to the deterministic calculation of r, a specific message produces the same signature independent of how often the message is transmitted. If the same message is transmitted twice and r has not been derived from the messages but carries (two distinct) subliminal messages, this can be detected by the fact that the signatures differ although the messages are identical. A warden who monitors the communication can notice that two signatures are different although they were generated for identical messages using the same key pair. From this observation,

Table 3: Approaches to ensure subliminal-freeness in EdDSA.

Approach	Applicability	Advantages and drawbacks
Pre-published nonce points	DSA-like and Schnorr signatures	+ Simple + Low computational requirements – Limited number of transmitted messages – Subliminal information embeddable during list computation – Storage requirements for warden
Warden interaction [29]	Schnorr signatures	+ Small bandwidth requirements – Participation of warden required – Several messages need to be exchanged – Need for bidirectional communication – Subliminal channel to/from warden possible
Zero-knowledge proofs [6]	Proving pseudorandomness of a curve point	+ Simple communication pattern + Feasible for offline scenarios – Huge prove size – Significant computational requirements – Subliminal channel to warden possible

the warden may then deduce that subliminal information has been transferred. In order to prevent this a subliminal sender would need to check if an identical message has been sent before. If this is the case, the same value r should be used to not raise suspicion. The subliminal receiver can just discard any subliminal information received in duplicated messages. Obviously, this method increases storage requirements for both, subliminal sender and subliminal receiver, significantly.

7.2.2 Small Nonce Values. As described in Section 3, it is of utmost importance for the security of the signature system to sustain unpredictability of the nonce value r. However, when directly encoding the (unencrypted) subliminal information into r, it may regularly take small values or even become equal to zero, depending on the subliminal information that is being transmitted. Detection of such values can, therefore, not only lead to detection of the subliminal channel, but also allow an eavesdropper to recover the signing key a. The sender of the subliminal information can mitigate this problem, though, by encrypting the subliminal information. In fact, encryption is often performed for covert and subliminal channels to prevent others from being able to read the transferred information in case the channel is detected. It is important to note, however, that unlike signature schemes that consume true randomness, encryption in this case cannot prevent detection of the subliminal channel in case that someone knows the secret key k (e.g., in a special administratory position). In this case the secret key k together with the message can be used to verify if r has been computed conforming to Eq. (2).

7.2.3 Repeating Nonce Values. As explained in Section 3, the signing key can be recovered from signatures with distinct messages which are using the same nonce value r. The sender of the subliminal information therefore has to ensure that this case does not occur if r is used for subliminal information instead of calculating it from the message. Depending on the type of information that is to be transmitted covertly, repeating values of r might occur especially if naive encryption methods are used. To counter this problem, the initialization vector for encryption of the covert information should be derived from the overt message in a manner similar to Eq. (2). Additionally, the output feedback (OFB) mode of block ciphers can be used to significantly reduce the probability of reoccurring random values [31].

8 CONCLUSION

In this paper, we conducted a theoretical analysis of EdDSA and identified a way to inject a subliminal channel in EdDSA signatures. The efficiency of the subliminal channel is substantial (nearly 50 %). We investigated several use case for EdDSA, including new application areas for high-speed signatures like clock synchronization and smart grid sensor data collection, as well as classical and recent versions of TLS. We conducted experimental measurements for these scenarios in order to show the applicability of the subliminal channel and measure the actual bandwidth of information that can be leaked in different use cases through the subliminal channel. We then validated existing countermeasures against subliminal channels in other signature schemes for their applicability to EdDSA and found that none of the countermeasures is viable in the context of network security. We also proposed some detection methods, but those also can be circumvented by careful adversaries.

Given the fact that EdDSA is already used in network security protocols and its use is likely to increase substantially due to its appealing properties (high performance and small signature size), the existence of a subliminal channel is a significant threat to information security in many scenarios. We conclude that when employing EdDSA in current or future protocols network operators, security engineers, as well as protocol designers should be aware of the existence of the subliminal channel described in this paper and about the limitations of potential countermeasures. If a subliminal channel cannot be tolerated in the target scenario, either a different, subliminal-free signature scheme (with potentially less attractive properties) should be employed instead or significant resources are required to check that the subliminal channel is not actively exploited.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable suggestions.

REFERENCES

[1] Daniel J. Bernstein et al. "High-speed high-security signatures". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 124–142.

[2] Gustavus J. Simmons. "The Prisoners' Problem and the Subliminal Channel". en. In: *Advances in Cryptology*. DOI: 10.1007/978-1-4684-4730-9_5. Springer US, 1984, pp. 51–67. ISBN: 978-1-4684-4732-3 978-1-4684-4730-9.

[3] Gustavus J. Simmons. "Subliminal Communication is Easy Using the DSA". In: *Advances in Cryptology — EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus*. Springer Berlin Heidelberg, 1994, pp. 218–232. ISBN: 978-3-540-48285-7. DOI: 10.1007/3-540-48285-7_18.

[4] Ross Anderson et al. "The Newton channel". In: *Information Hiding: First International Workshop*. Springer Berlin Heidelberg, 1996, pp. 151–156. ISBN: 978-3-540-49589-5. DOI: 10.1007/3-540-61996-8_38. URL: https://doi.org/10.1007/3-540-61996-8_38.

[5] Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *Advances in cryptology*. Springer. 1985, pp. 10–18.

[6] Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. "A subliminal-free variant of ECDSA". In: *International Workshop on Information Hiding*. Springer, 2006, pp. 375–387.

[7] Q. Dong and G. Xiao. "A Subliminal-Free Variant of ECDSA Using Interactive Protocol". In: *2010 International Conference on E-Product E-Service and E-Entertainment*. Nov. 2010, pp. 1–3. DOI: 10.1109/ICEEE.2010.5660874.

[8] Don Johnson, Alfred Menezes, and Scott Vanstone. "The elliptic curve digital signature algorithm (ECDSA)". In: *International Journal of Information Security* 1.1 (2001), pp. 36–63.

[9] Xianfeng Zhao and Ning Li. "Reversible Watermarking with Subliminal Channel". In: *Information Hiding: 10th International Workshop, IH 2008*. Springer Berlin Heidelberg, 2008, pp. 118–131. ISBN: 978-3-540-88961-8. DOI: 10.1007/978-3-540-88961-8_9.

[10] Jens-Matthias Bohli and Rainer Steinwandt. "On Subliminal Channels in Deterministic Signature Schemes". In: *7th International Conference on Information Security and Cryptology – ICISC 2004*. Springer Berlin Heidelberg, 2005, pp. 182–194. ISBN: 978-3-540-32083-8. DOI: 10.1007/11496618_14.

[11] Ronald L. Rivest, Adi Shamir, and Len Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[12] Daniel J. Bernstein. "Curve25519: New Diffie-Hellman Speed Records". In: *9th International Conference on Theory and Practice in Public-Key Cryptography (PKC 2006)*. Springer Berlin Heidelberg, 2006, pp. 207–228. ISBN: 978-3-540-33852-9. DOI: 10.1007/11745853_14.

[13] S. Josefsson and I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032 (Informational). Internet Engineering Task Force, Jan. 2017. URL: http://www.ietf.org/rfc/rfc8032.txt.

[14] Mike Hamburg. *Ed448-Goldilocks, a new elliptic curve*. Cryptology ePrint Archive, Report 2015/625. http://eprint.iacr.org/2015/625. 2015.

[15] Claus P. Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology - CRYPTO '89*. New York: Springer, 1990, pp. 239–252.

[16] Robert Annessi, Joachim Fabini, and Tanja Zseby. *SecureTime: Secure Multicast Time Synchronization*. 2017. eprint: arXiv:1705.10669.

[17] E. Itkin and A. Wool. "A security analysis and revised security extension for the precision time protocol". In: IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS). Sept. 2016, pp. 1–6. DOI: 10.1109/ISPCS.2016.7579501.

[18] Eyal Itkin and Avishai Wool. *A Security Analysis and Revised Security Extension for the Precision Time Protocol*. 2016. eprint: arXiv:1603.00707.

[19] Ik Rae Jeong et al. "Provably Secure Encrypt-then-Sign Composition in Hybrid Signcryption". In: *Information Security and Cryptology — ICISC 2002: 5th International Conference Seoul, Korea, November 28–29, 2002 Revised Papers*. Ed. by Pil Joong Lee and Chae Hoon Lim. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 16–34. ISBN: 978-3-540-36552-5. DOI: 10.1007/3-540-36552-4_2. URL: https://doi.org/10.1007/3-540-36552-4_2.

[20] Yoav Nir, Simon Josefsson, and Manuel Pegourie-Gonnard. *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier*. Internet-Draft draft-ietf-tls-rfc4492bis-17. http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc4492bis-17.txt. IETF Secretariat, May 2017. URL: http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc4492bis-17.txt.

[21] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. http://www.rfc-editor.org/rfc/rfc5246.txt. RFC Editor, Aug. 2008. URL: http://www.rfc-editor.org/rfc/rfc5246.txt.

[22] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Internet-Draft draft-ietf-tls-tls13-20. http://www.ietf.org/internet-drafts/draft-ietf-tls-tls13-20.txt. IETF Secretariat, Apr. 2017. URL: http://www.ietf.org/internet-drafts/draft-ietf-tls-tls13-20.txt.

[23] Eu-Jin Goh et al. "The Design and Implementation of Protocol-Based Hidden Key Recovery". en. In: *Information Security*. Lecture Notes in Computer Science 2851. DOI: 10.1007/10958513_13. Springer Berlin Heidelberg, Oct. 2003, pp. 165–179. ISBN: 978-3-540-20176-2 978-3-540-39981-0.

[24] Justin Merrill and Daryl Johnson. “Covert Channels in SSL Session Negotiation Headers”. In: *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015, p. 70.

[25] Carlos Scott. *Network covert channels: Review of current state and analysis of viability of the use of x. 509 certificates for covert communications.* Tech. rep. RHUL-MA-2008-11, Department of Mathematics, Roal Holloway, University of London (January 2008), 2008.

[26] Zbigniew Gołębiewski, Mirosław Kutyłowski, and Filip Zagórski. “Stealing secrets with SSL/TLS and SSH – Kleptographic attacks”. In: *International Conference on Cryptology and Network Security*. Springer, 2006, pp. 191–202.

[27] Adam L. Young and Moti M. Yung. “Space-Efficient Kleptography Without Random Oracles”. en. In: *Information Hiding*. DOI: 10.1007/978-3-540-77370-2_8. Springer Berlin Heidelberg, June 2007, pp. 112–129.

[28] O. Sury and R. Edmonds. *Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC*. RFC 8080 (Proposed Standard). Internet Engineering Task Force, Feb. 2017. URL: http://www.ietf.org/rfc/rfc8080.txt.

[29] Yinghui Zhang et al. “Provably secure and subliminal-free variant of schnorr signature”. In: *Information and communication technology-EurAsia conference*. Springer, 2013, pp. 383–391.

[30] M. Naor and O. Reingold. “Number-theoretic constructions of efficient pseudo-random functions”. In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. Oct. 1997, pp. 458–467. DOI: 10.1109/SFCS.1997.646134.

[31] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.