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. Therefore the usability for leaking information is limited and the existence of the subliminal channel may be tolerable in some scenarios. However, with the advent of high-speed signatures, such as EdDSA, now 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 a large number of 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 classical 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 that are secured by EdDSA.

CSC CONCEPTS
• Security and privacy → Digital signatures; Security protocols;
• Networks → Network protocols; • Social and professional topics → Censorship; Computer crime; • Hardware → Smart grid;

*Corresponding author.

KEYWORDS
Censorship circumvention, cyber-physical systems, data exfiltration, digital signatures, ed25519, EdDSA, information hiding, information leakage, network protocols, network security, subliminal channels

1 INTRODUCTION
Subliminal channels are used to hide information in cryptographic protocols and can, therefore, be used to clandestinely leak information. A typical use of subliminal channels in digital signatures are situations where encryption is not permitted or would raise suspicion 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, typical censorship scenarios, or scenarios in which encryption is just unusual.

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 time or 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, when confidential information is leaked to the outside.

In classical security protocols, such as IPsec or TLS, digital signatures are usually used only sparsely, mainly in the authentication phase. With this, the bandwidth for data leakage is bound to a few bytes per connection. A subliminal channel in a signature scheme in a security protocol may be sufficient to leak critical information such as keying material or status information but has a quite limited bandwidth.

The advent of high-speed signatures such as EdDSA [1] does change this. EdDSA was primarily designed for small signature size as well as high-performance, i.e., fast signing and verification. With those properties it can be used in scenarios where many subsequent packets need to be signed, like clock synchronization 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 analyse what potential bandwidth 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 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 Transport Layer Security (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).

In [3], Simmons shows 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 that was shown by Anderson et al. for the ElGamal signature scheme specifically [4]. Here, the signer reveals as many bits of information of the secret key to the subliminal receiver as should be used for 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 that the randomness, which is essential to the algorithm, is (mis)used in this way. 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

\[ E = \{ (x, y) \in F_p \times F_p : -x^2 + y^2 = 1 + dx^2y^2 \} . \] (1)

where \( F_p \) 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 \( 2^cL \) equals the number of points on \( E \). Furthermore, a hash function \( H \) is used, which produces \( 2b \) bit output, where \( b \in \mathbb{N} \) determines the security level that is achieved by the signature scheme. 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 $b$ bits. It is mapped to a $2b$-bit string $h = H(k)$. Bits $h_0$ to $h_{n-1}$ with $c \leq n < b$ 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

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

(2)

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

$$2^S SB = 2^r R + 2^r H(R,A,M)A.$$  

(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 $a = (S - r)/H(R,A,M) \mod L$ 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 $a = (S_1 - S_2)/(H(R,A,M_1) - H(R,A,M_2)) \mod L$. 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 the 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

$$r = S - H(R,A,M)a \mod L$$

(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.

Please note that if data is processed with the encrypt-then-sign method the message $M$ is the ciphertext and not the plaintext. This 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 $2^R$ values for $r$ on average, where $B_r$ denotes
the desired bandwidth of the subliminal channel in bits. Due to the
exponential growth with bandwidth, it is not possible to use a signif-
icant 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 have to possess 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 communi-
cation 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 subsequent packets and
needs to ensure integrity and data origin authentication to the
receiver(s) [16, 17, 18]. EdDSA is an excellent choice for such sce-
narios as it is fast and lightweight enough to be applied in 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 about 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
encrypted. 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 con-
identiality so that data may be sent unencrypted (or would use the
encrypt-then-sign method). Those use cases directly correspond to
the classical prison scenario in the original work by Simmons 1984.
In addition, we show as third use case examples for the use of sub-
liminal channels in TLS where in some cases the signature is part
of the encrypted information.

\(^1\)For a (long but still incomplete) list of projects that employ EdDSA see https://ianix.
com/pub/ed25519-deployment.html.

As exemplary use cases we investigate the following scenarios:

- **Broadcast clock synchronization**: Clock synchronization
  protocols, such as Network Time Protocol (NTP) and Preci-
sion Time Protocol (PTP) can use high-speed signatures for
  signing messages to provide data origin authentication.
  Time synchronization require data origin authentication
  to ensure that the time information comes from a trusted
  server and has not been modified on the path. Confiden-
tiality 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 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 collec-
tors in smart grid monitoring. Depending on the amount
of sensor data, many signatures need to be sent in short
  time intervals. Again here 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 synchroniza-
tion 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 func-
tioning of security protocols that rely on accurate time informa-
tion [16].

EdDSA seems particularly well-suited for this purpose as it pro-
vides adequate performance in terms of signing and verification
speed and achieves good security properties. However, the possi-
bility of embedding subliminal information in the signatures has
to be taken into account in security-sensitive environments. A sig-
nature 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 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 speed and resource challenges. 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 measurements values per second are transmitted to data concentrators and may need to be processed 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.

For security protocols we need to distinguish between covert channels and subliminal channels. Covert channels use protocol fields in the header or packet timing to exchange covert information. Some work has been done to identify covert channels in TLS. 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, Kutylowski, 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 view show a bit pattern that corresponds to the subliminal information.

5.3.2 Diffie-Hellman parameters. One possibility similar to the RSA approach described by Gołąbiewski, Kutylowski, 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 $a$ a value of $g^a$ that shows the desired bit pattern can finally be found.

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 $g^a$. The discrete logarithm $a$ has to be known by the sender in order to calculate the joint secret key. If a sender wants to transmit information directly in $g^a$ he would need to find an $a$ that would generate the subliminal message $g^a$. Thus, since computing the discrete logarithm is infeasible, these points $g^a$ cannot be chosen equal to the intended subliminal information.

5.3.3 Signatures. In contrast to the two subliminal channels methods 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 therefore the signed message (which is needed to recover subliminal information in the signature) is known by anyone listening on the network. Furthermore, as described in section 7.2 on page 20, 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, in the upcoming version TLS 1.3 the use of ephemeral Diffie-Hellman is enforced 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 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. In this case an eavesdropper cannot see the signed message and therefore could not recover the subliminal information in $r$. So in order to use this as subliminal channel the subliminal receiver would also need 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 facing page provides references for subliminal channels discovered in signature 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 [29]. 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

Besides the attack scenarios already investigated in this section, further scenarios for the use of the subliminal channel in EdDSA are possible 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 provide additional information without the owner’s knowledge. Also,
Table 1: Subliminal channels usable in TLS.

<table>
<thead>
<tr>
<th>Exploited scheme</th>
<th>Protocol version</th>
<th>Bandwidth</th>
<th>Difficulty of detection</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key exchange using RSA</td>
<td>≤ TLS 1.2</td>
<td>narrow/broad</td>
<td>hard</td>
<td>[26]</td>
</tr>
<tr>
<td>Diffie-Hellman key exchange</td>
<td>all</td>
<td>narrow</td>
<td>hard</td>
<td></td>
</tr>
<tr>
<td>DSA</td>
<td>≤ TLS 1.2</td>
<td>narrow/broad</td>
<td>hard</td>
<td>[3]</td>
</tr>
<tr>
<td>ECDSA</td>
<td>all</td>
<td>narrow/broad</td>
<td>hard</td>
<td>[6]</td>
</tr>
<tr>
<td>EdDSA</td>
<td>all</td>
<td>narrow/broad</td>
<td>hard</td>
<td>this paper</td>
</tr>
</tbody>
</table>

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.

DNSCrypt \(^2\) uses Ed25519 and DNSSEC \(^2\) supports it, which is why these protocols are susceptible to subliminal channels. Furthermore, upcoming efforts to secure Internet Inter-AS routing in the Border Gateway Protocol (BGP) use nested signatures for path validation. In these signatures, subliminal information could be transmitted between BGP routers. 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.](https://example.com/figure4.png)

To verify the scenario described above the following experimental setup was used (Fig. 4). The NTP server and client run unmodified NTP software\(^3\) 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 a typical setup required 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 \(^4\) 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

\(^2\)https://dnscrypt.org/

\(^3\)The NTP server and NTP client run NTP version 4.2.6.p5 (but we do assume our experiment to not dependent on specific NTP versions, since we run unmodified NTP code and do not rely on any special features of NTP).

\(^4\)https://nacl.cr.yp.to/
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 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 webserver and a simple HTTP client application, both compiled with Google’s OpenSSL fork BoringSSL 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 webservice 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.

Considering TLS 1.2, an example of information that could be transmitted using this subliminal channel is keying material, which allows a passive eavesdropper to decrypt the data that is meant to be protected by TLS. A different scenario where it could be useful to an attacker 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.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTP broadcasts</td>
<td>max. 3.9 B/s</td>
</tr>
<tr>
<td>PMU measurements</td>
<td>310 B/s</td>
</tr>
<tr>
<td>TLS handshake</td>
<td>bidirectionally 31 B</td>
</tr>
</tbody>
</table>

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.

7 APPROACHES TO MITIGATE SUBLIMINAL COMMUNICATION

Some approaches to mitigate or at least to detect subliminal communication have been published in the literature for other signature schemes. In this section, we investigate if any of these methods can be applied to protect EdDSA signatures against the establishment of subliminal channels.

In order to elaborate on mitigation possibilities, we introduce a ’warden’ to the subliminal scenario established in Section 4 (see Fig. 5). The warden is a trusted instance that is located between signer and verifier(s) 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 prisoners need 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.

7.1 Ensuring Subliminal-Free Signatures

In some situations it is possible to ensure that no subliminal information is embedded in a signature. To this end, 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.

The logical instance of a warden may also be co-located with the verifier.

http://nginx.org/
https://boringssl.googlesource.com/boringssl/

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.
clandestinely during the signing process. Then, during signature generation, the signer has to use the values in the same order as they appear in the list, leaving just one possible value for $S$ for a given message. A variant of this approach would be to index the list by a number derived deterministically from the message. This eliminates the need for keeping a state for signature generation. However, it causes an increasing number of messages to be unsignable as they would require a nonce that was already used earlier and would therefore make the signature reveal the signing key.

Both approaches have 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 approach, 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 would need to store the list of $R$-values, which may lead to significant storage requirements. For each message the signer wants to possibly 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 would just be 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. 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 a signature scheme cannot be made subliminal-free (without introducing serious disadvantages), there may still exist ways to detect the subliminal data transfer or at least to introduce some checks to see if subliminal data exchange may be suspected. In this section, we highlight situations in which the subliminal channel can lead to observable suspicious patterns that may help an observer to detect the transmission of subliminal information.

#### 7.2.1 Equal Messages

Due to the deterministic calculation of $r$ based on the message, 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 all sent packets
can notice that two signatures are different although they were generated for identical messages using the same key pair. From this observation, 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 equal message has been sent before. If this is the case, he should simply use the same value \( r \) to not raise suspicion. The subliminal receiver can just discard any subliminal information that he 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 \( r \). Therefore, the sender of the subliminal information has to ensure that this case does not occur if he uses \( r \) 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.

REFERENCES


