Aalto University

The Undeniable Truth:

How Remote Attestation Circumvents Deniability Guarantees in Secure Messaging Protocols

Lachlan J. Gunn, Ricardo Vieitez Parra, N. Asokan,
Aalto University

Two trends

1. Hardware-based trusted execution environments (TEEs) are pervasively deployed

2. Increasing popularity of deniable communication mechanisms in messaging apps

Outline

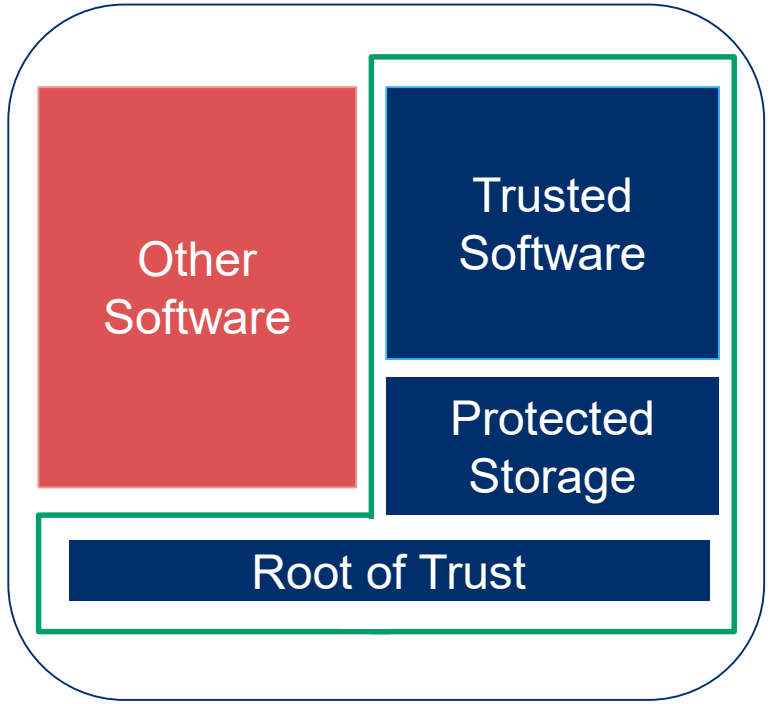
- What are TEEs and remote attestation
- What is deniability and why should we care?

- Attack: Breaking deniability of messaging protocols using remote attestation
- Demo
- Countermeasures

- The Big Picture

TEEs and Attestation

Hardware-security mechanisms are pervasive



Hardware support for

- **Isolated execution: Isolated Execution Environment**
- **Protected storage: Sealing**
- **Ability to report status to a remote verifier: Remote Attestation**

Trusted Execution Environments (TEEs)

Cryptocards

https://www.ibm.com/security/cryptocards/

Trusted Platform Modules

https://www.infineon.com/tpm

ARM TrustZone

https://www.arm.com/products/security-on-arm/trustzone

Intel Software Guard Extensions

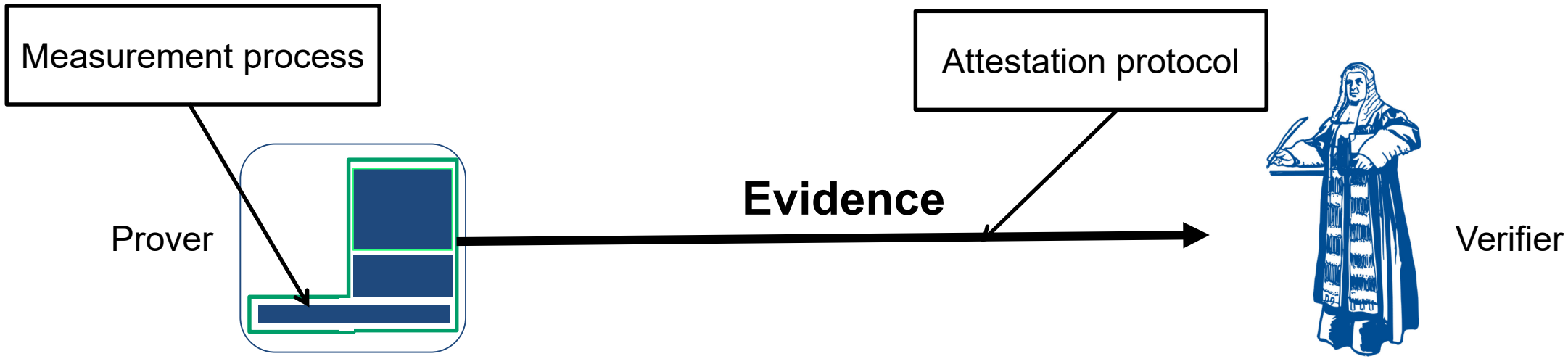
https://software.intel.com/en-us/sgx

[A+14] "Mobile Trusted Computing", Proceedings of the IEEE, 102(8) (2014)
[EKA14] "Untapped potential of trusted execution environments", IEEE S&P Magazine, 12:04 (2014)

What is remote attestation?

Verifier **ascertains current state and/or behaviour of** ***Prover***

What are the security requirements?



Attestation requirements

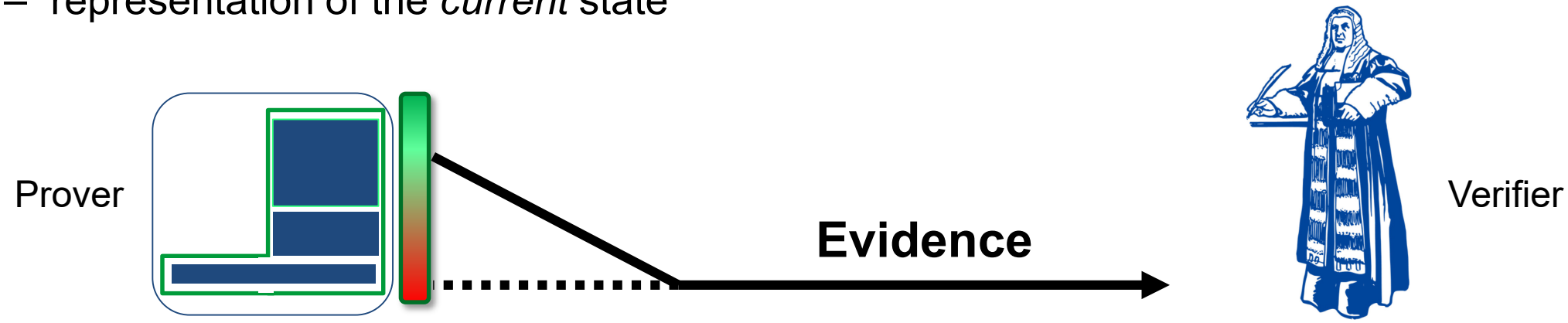
1. Authenticity
   – representation of the *real* state of the system

Need: signed evidence, a certified key unique to the device (trust in the CA), a root of trust on device.

Attestation requirements

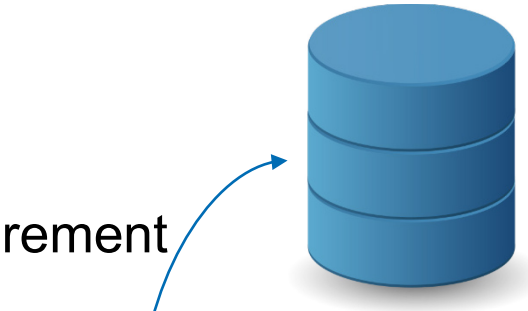
1. Authenticity
   - representation of the *real* state of the system

2. “Timeliness”
   - representation of the *current* state



Need: signed evidence, a certified key unique to the device (trust in the CA), a root of trust on device.
Need: fresh nonce from verifier included in signed evidence

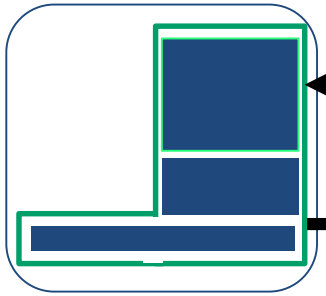
Attestation Protocol



5. Verify measurement

Database of acceptable measurements

1. securely measure and store *measurement*



Prover

2. nonce: n_1

4. quote, $certificate_{AK}$

Verifier

3. quote = $signature_{AK}$(*measurement*, n_1 || channel info)

AK: attestation key known only to root-of-trust on device

$Certificate_{AK}$: certificate for AK issued by a CA trusted by verifier

Attestation in practice

Introduced in late 1990s by Trusted Computing Group for Trusted Platform Modules

Supported in modern TEEs (Intel SGX, certain Trusted OSs for ARM TrustZone)

Measurement: hash of executable ("binary attestation"); can be of arbitrary property

Attestation can be chained

- Binary attestation to verify some application (and its key) and some application-provided data
- Property attestation verified by application and signed by application key

If your TEE can locally verify some property, it can convince a remote verifier of the same

Deniable Messaging

Desiderata for messaging protocols

1. **Authenticity.** If I send you a message, you can tell whether it is authentic

Deniable protocols have an extra, seemingly conflicting objective:

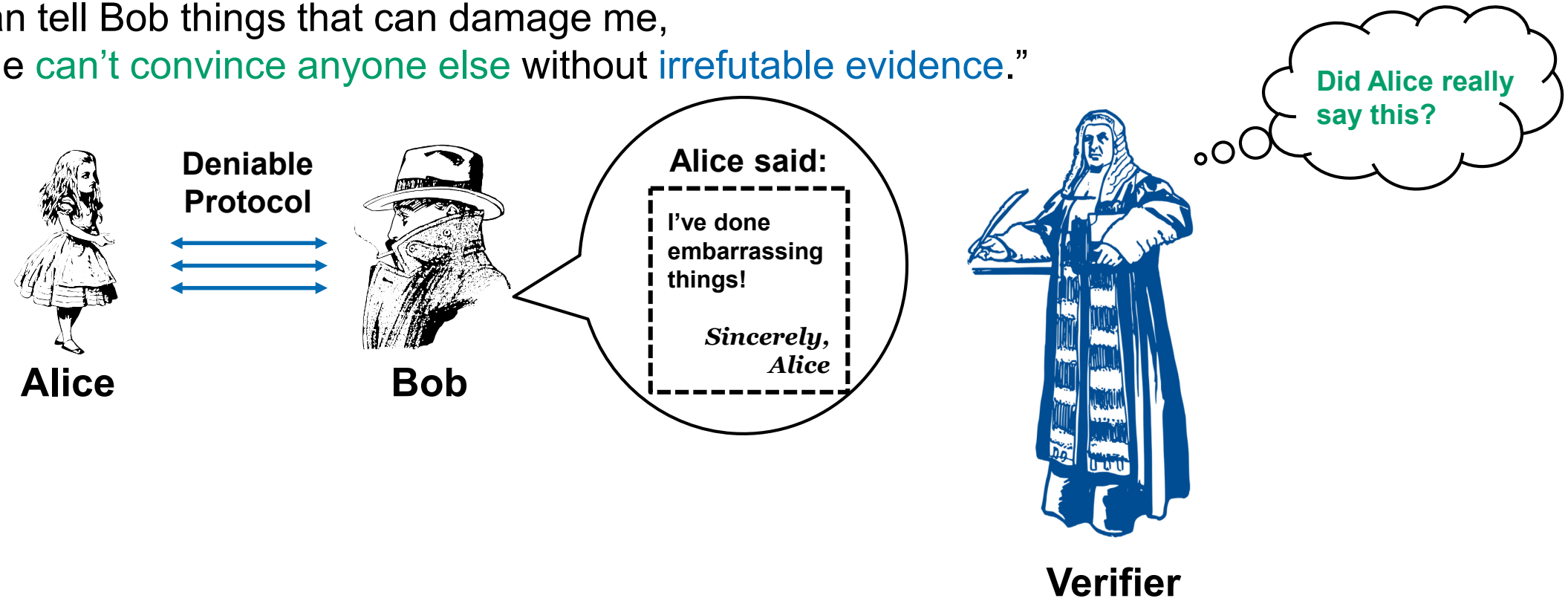
2. **Deniability.** You can't prove to anyone else that a message came from me
   - Recipient can differentiate between real messages and forgeries
   - Goal: easy to make forgeries that look realistic to everyone else

Signal, WhatsApp, Pidgin etc. now include protocols for cryptographic deniability

The limits of deniability?

A naïve view:

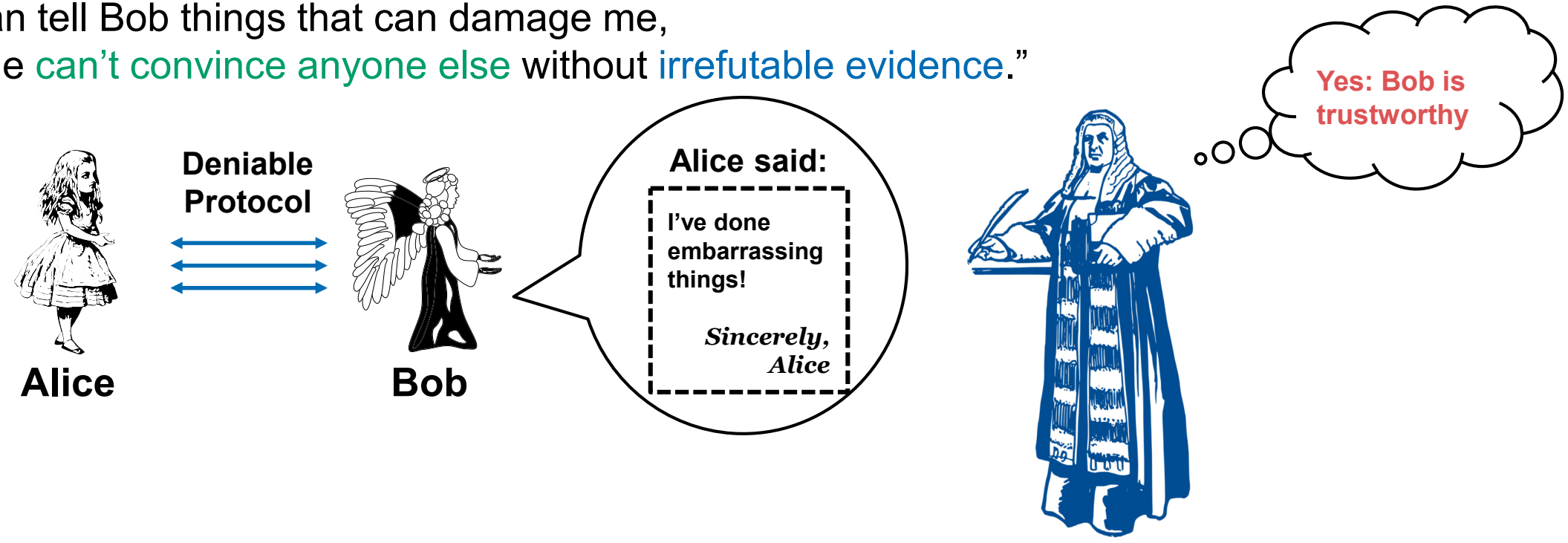
- Alice: “I can tell Bob things that can damage me, because he can’t convince anyone else without irrefutable evidence.”



The limits of deniability?

A naïve view:

- Alice: “I can tell Bob things that can damage me, because he can’t convince anyone else without irrefutable evidence.”



The reality:

- Verifiers don’t necessarily need irrefutable evidence
- Plaintext is enough if conveyed by a trusted informant

When is (cryptographic) deniability useful?

When the informant is untrustworthy 

People may trust: 

- Witnesses under oath
- Journalists
- …

But do you trust APT28? a.k.a.

- Fancy Bear
- Sofacy
- Guccifer 2.0
- GRU Units 26165/74455

A new kind of attack

Data dumps are now common:

- World Anti-Doping Agency (2016)
- Democratic National Committee (2016)
- En Marche (2017)
- Yousef Al Otaiba (2017)
- International Olympic Committee (2018)

But can include fabricated material

- thus limits attacker credibility

The dangers of undeniable communications

But the material itself may contain proof of origin

After the DNC 2016 email leaks:

- Some claimed emails were doctored

https://www.foxnews.com/politics/dnc-boss-brazile-claims-wikileaks-emails-doctored-in-contentious-interview

"I have seen so many doctored emails. I have seen things that come from me at two in the morning that I don't even send”

The dangers of undeniable communications

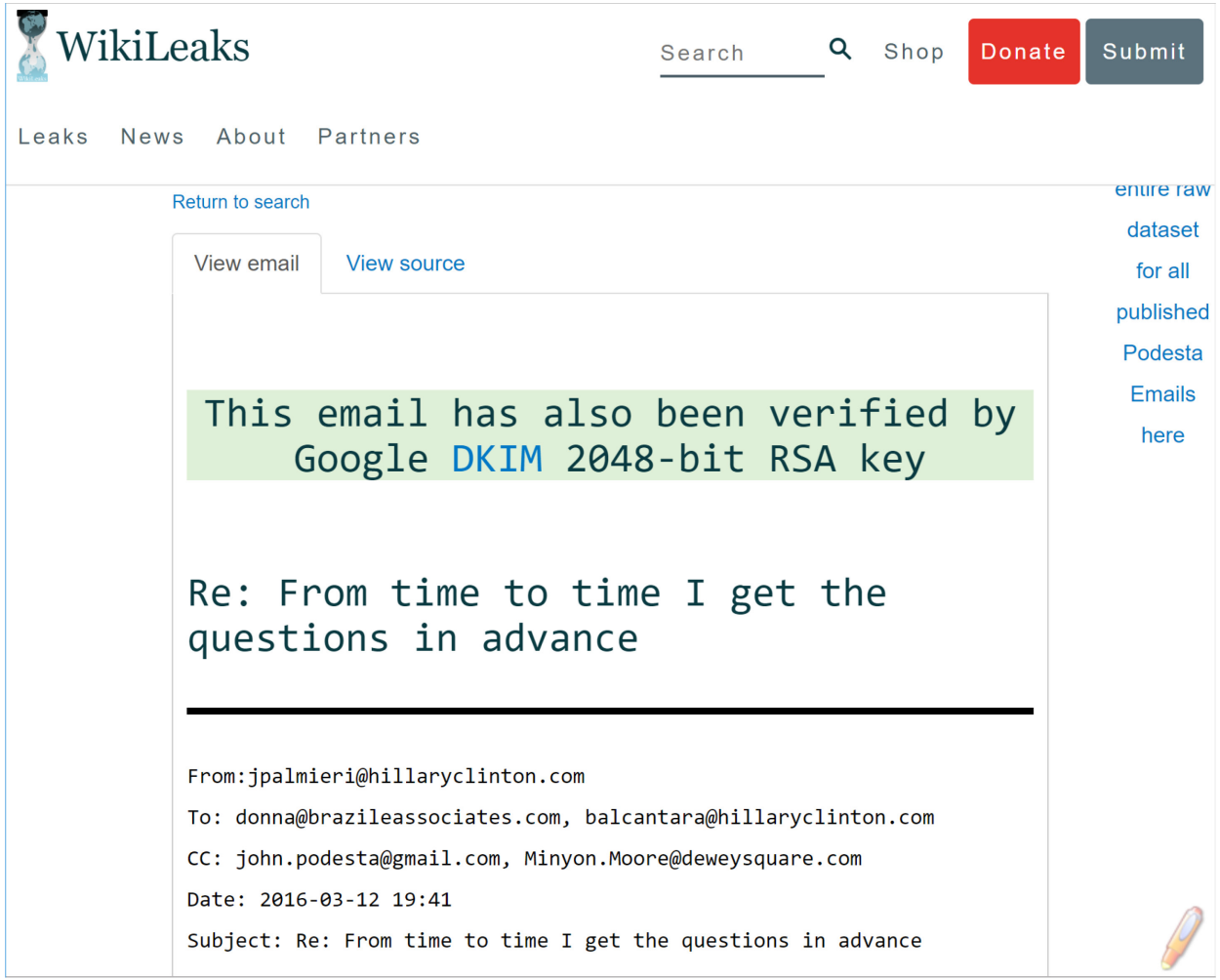
But the material itself may contain proof of origin

After the DNC 2016 email leaks:

- Some claimed emails were doctored

Shortly afterwards, WikiLeaks publish DKIM signatures

If you want deniability, you need to use deniable protocols



https://wikileaks.org/podesta-emails/emailid/5205

MOBILE

Signal encryption app sees 400 percent boost after election

The co-founder of Open Whisper Systems says installations of its app have increased four-fold since November 8.

https://www.cnet.com/news/signal-open-whisper-systems-donald-trump/

POLITICS

Messaging App Has Bipartisan Support Amid Hacking Concerns

Aides to Trump, Obama and de Blasio use Signal, a smartphone app that encrypts messages

By *Mara Gay*

Updated Jan. 24, 2017 11:16 a.m. ET

https://www.wsj.com/articles/messaging-app-has-bipartisan-support-amid-hacking-concerns-1485215028

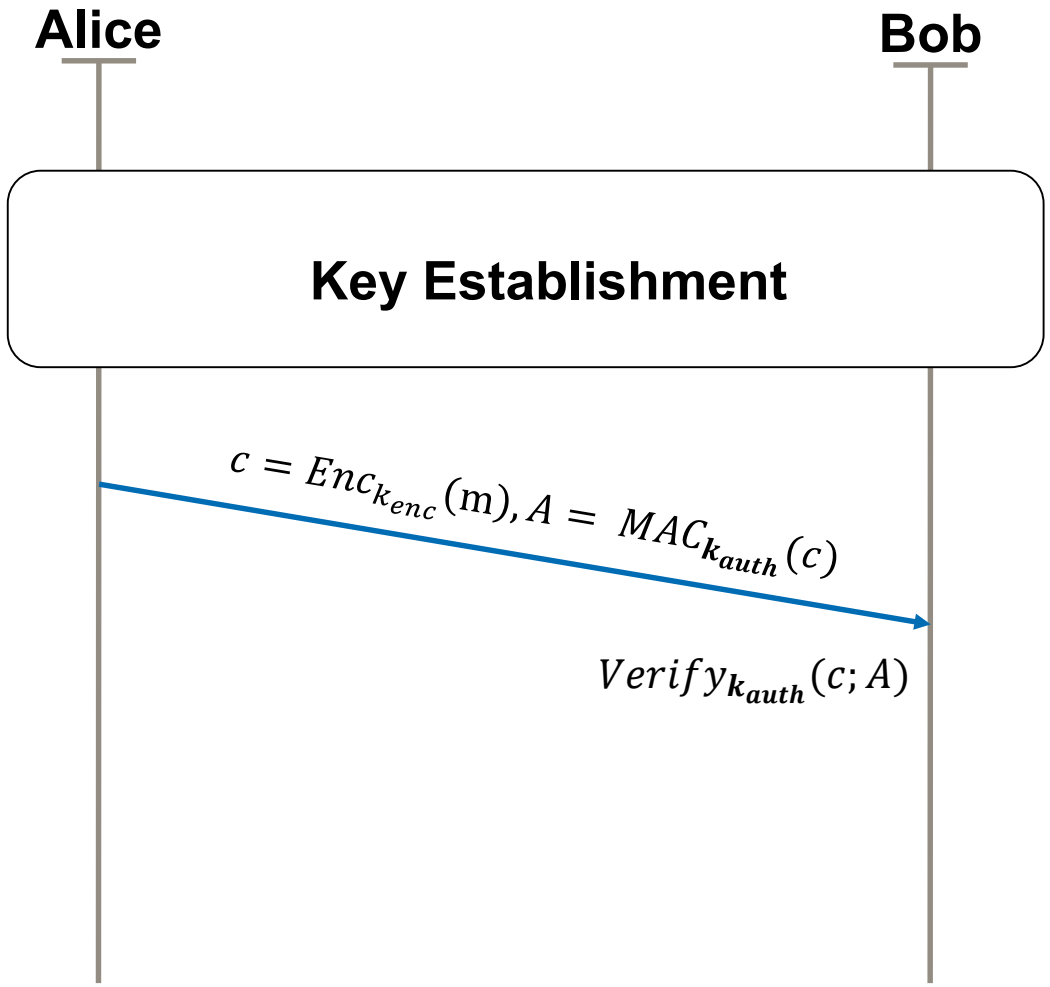
What do deniable protocols look like?

Asymmetric key-exchange protocol

- Result: shared symmetric key

Symmetric session crypto

- Verifying MAC requires the same key
- Able to verify ⇒ Able to forge



Easy to forge transcripts that look realistic

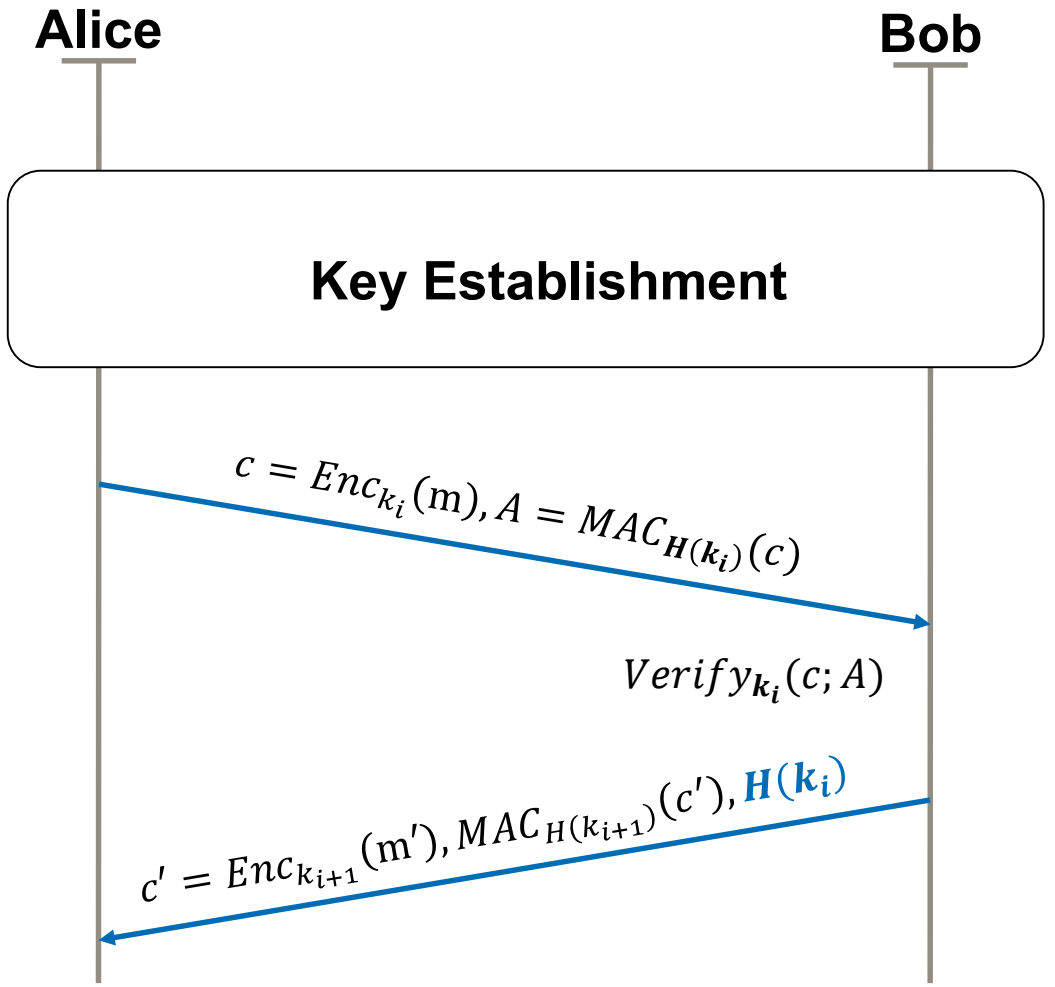
Deniable protocols: Off-the-Record (OTR)

First messaging protocol designed for deniability

Protocol flow:

1. Wait for message
2. Verify MAC on new message
3. Update MAC key; release previous MAC key

Anyone can now make valid authentication tags



Easy to forge transcripts that look realistic

Deniable protocols: Signal Protocol

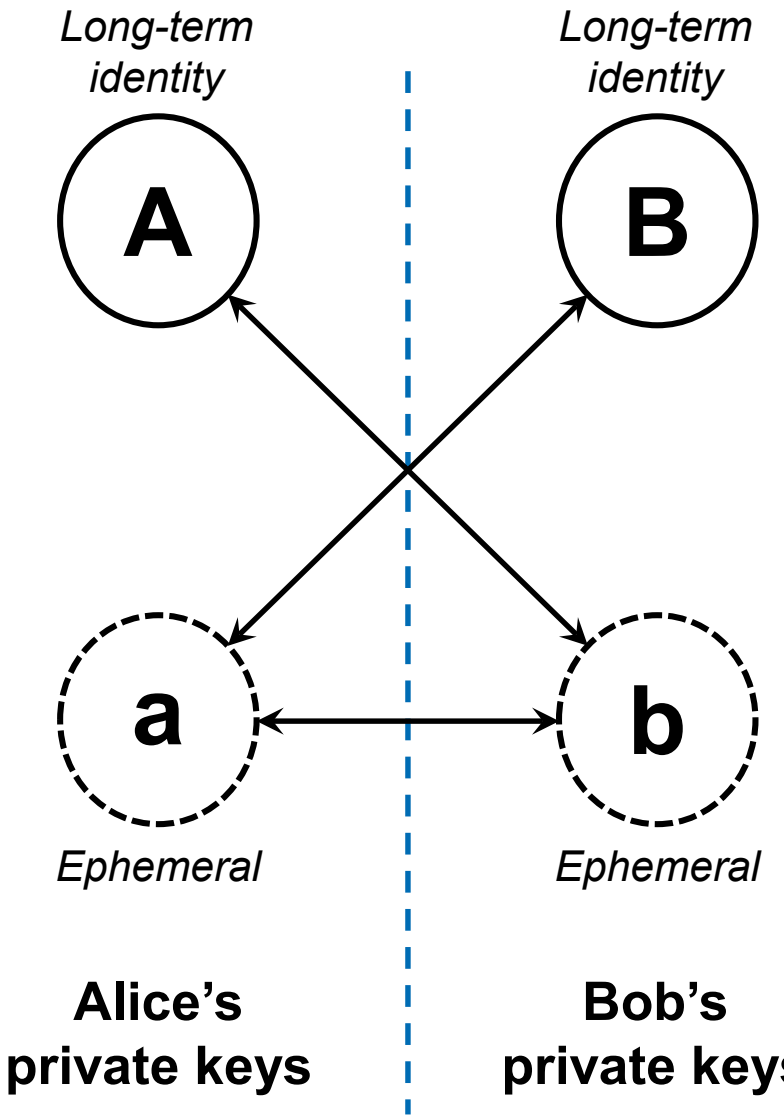
Key exchange: X3DH

$$k = H(\, g^{Ab} \parallel g^{aB} \parallel g^{ab} \,)$$

To get the key, need each Diffie-Hellman pair:

- A or b
- a or B
- a or b

If I know a and keep it secret, then I share the key with someone who knows B.

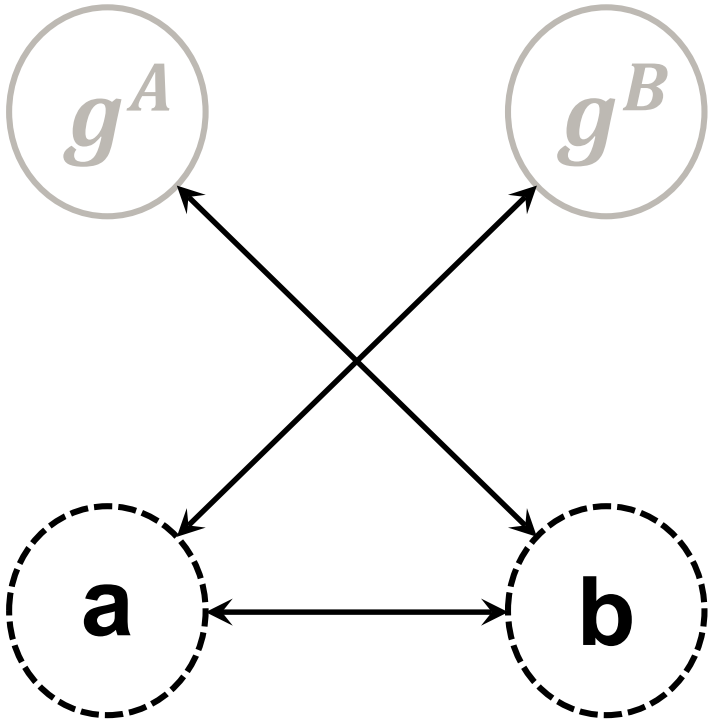


Deniable protocols: Signal Protocol

Anyone can forge the key exchange:

1. Pick random ephemeral private keys $\boldsymbol{a}$, $\boldsymbol{b}$
2. Look up public keys $\boldsymbol{g^A}$, $\boldsymbol{g^B}$
3. Compute $\boldsymbol{k}$ from *a*, *b*, and the public keys

But Bob can still authenticate Alice



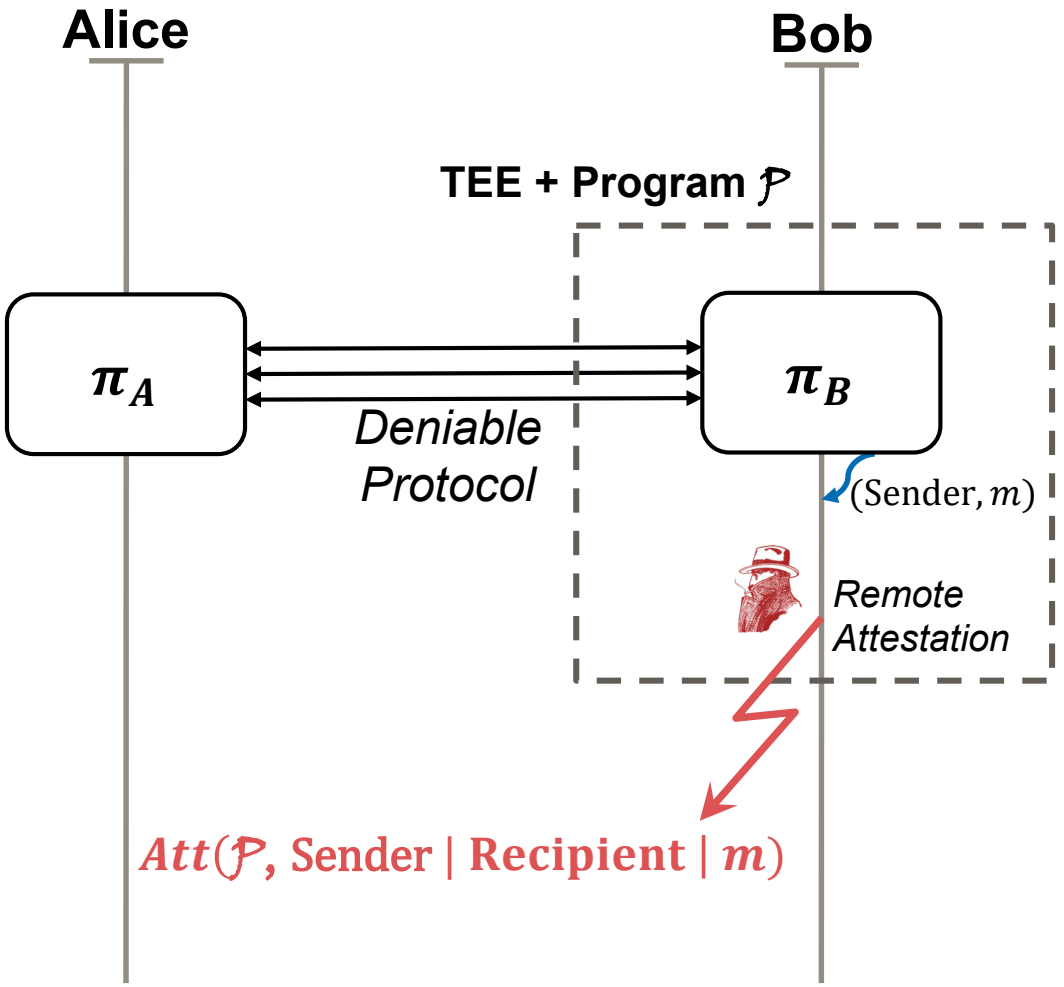
$$\boldsymbol{k = H(\, g^{Ab} \parallel g^{aB} \parallel g^{ab}\,)}$$

Easy to forge transcripts that look realistic

The Attack

An overview of our attack

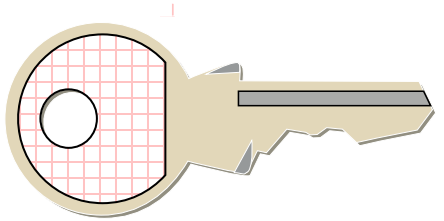
1. **Take a normal messaging client**
2. **Modify it to run inside a TEE**
3. **Produce a transcript of each session**
4. **Emit an attestation**
   - Shows that the transcript came from a correct client



Key point: TEEs let us prove that a key was secret

Symmetric authentication:

- Able to verify ⇒ Able to forge



No restrictions on usage.

Key point: TEEs let us prove that a key was secret

Symmetric authentication:

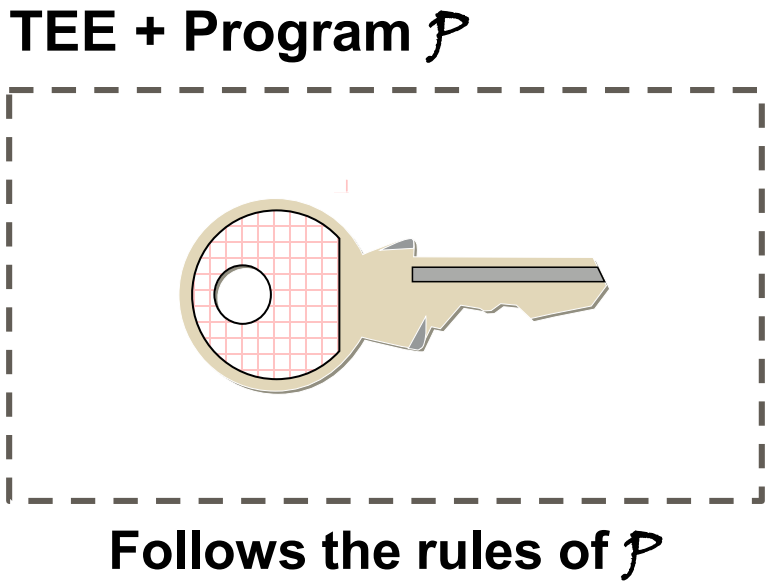
- Able to verify ⇒ Able to forge

A key in a TEE is protected.

- Only program $\mathcal{P}$ can use it

Remote attestation:

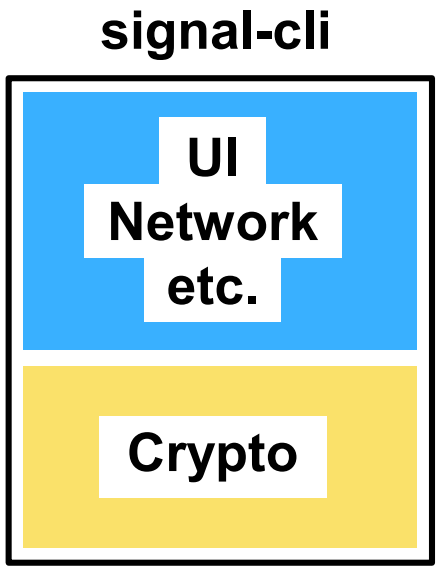
- Assures verifiers that TEE runs program $\mathcal{P}$
- Proof that Alice's messages in the transcript were not forged!



Modifying a Signal client

We use Signal as an example:

- Popular
- Convenient software architecture
- But any protocol would do



Modifying a Signal client

We use Signal as an example:

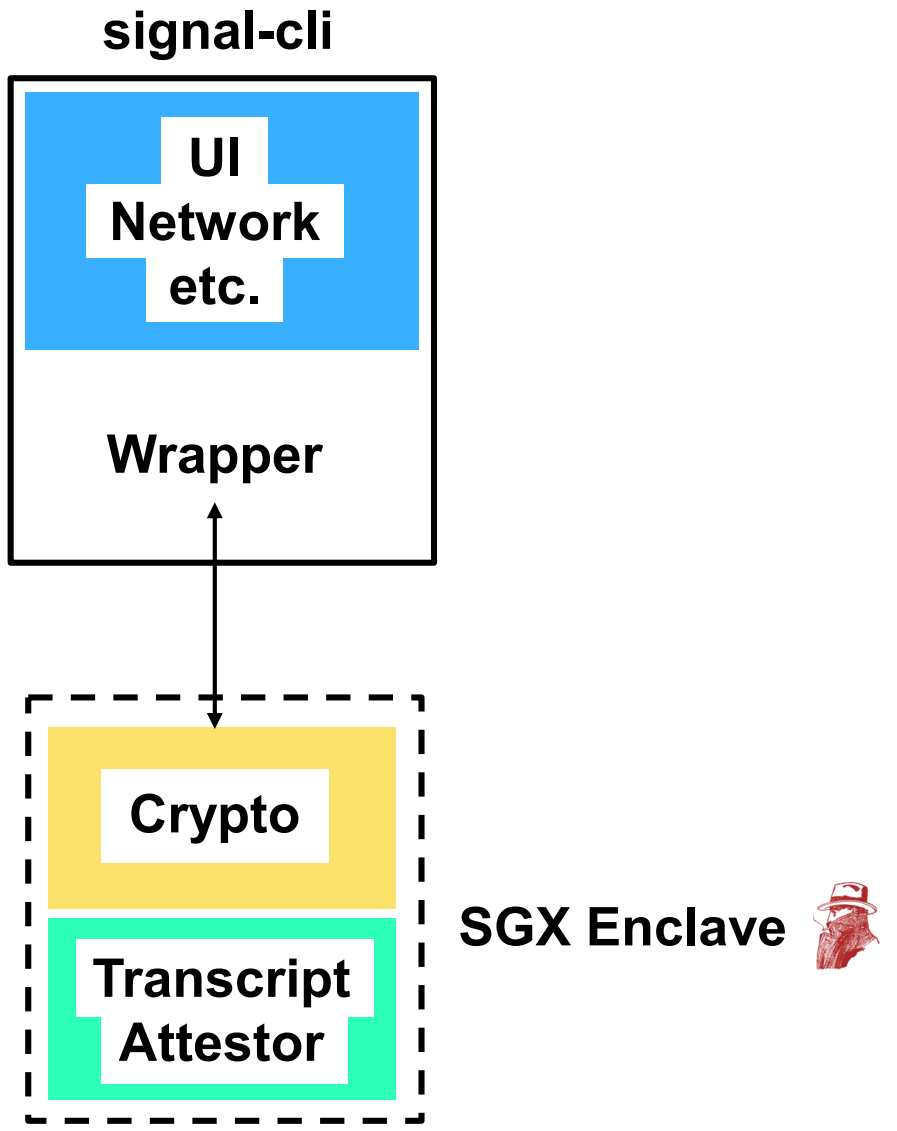
- Popular
- Convenient software architecture
- But any protocol would do

SGX enclave contains:

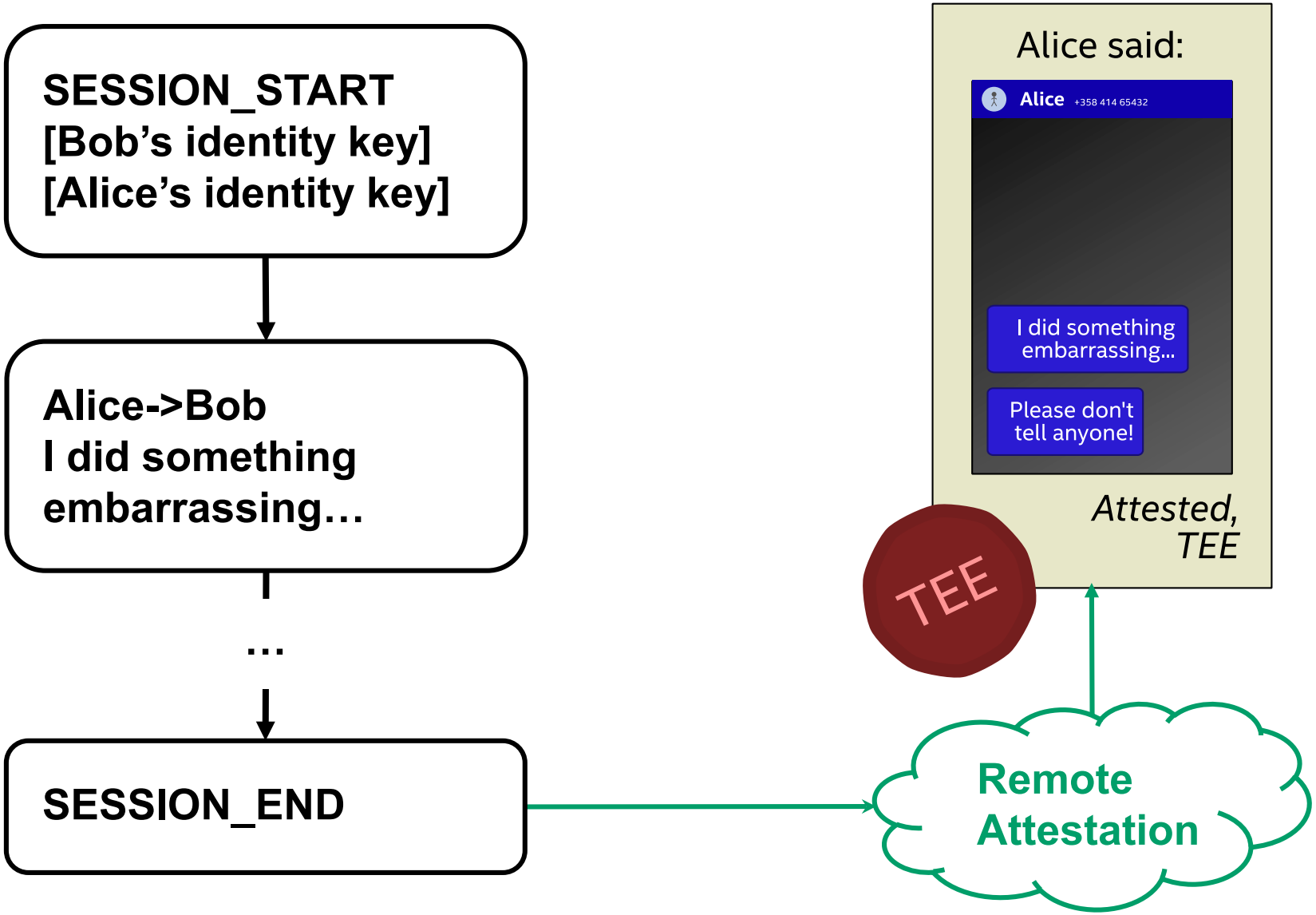
- *libsignal-protocol-c*
- Transcript generation

Modified (unofficial) *signal-cli*:

- Uses enclave for crypto



The result: an attested transcript



Demo

Countermeasures

Countermeasures

Skip to Big Picture

Switch to online-deniable protocols

Defensive remote attestation

Put the human in the loop

Switch to online-deniable protocols

“Classic” deniability fails with an interactive verifier

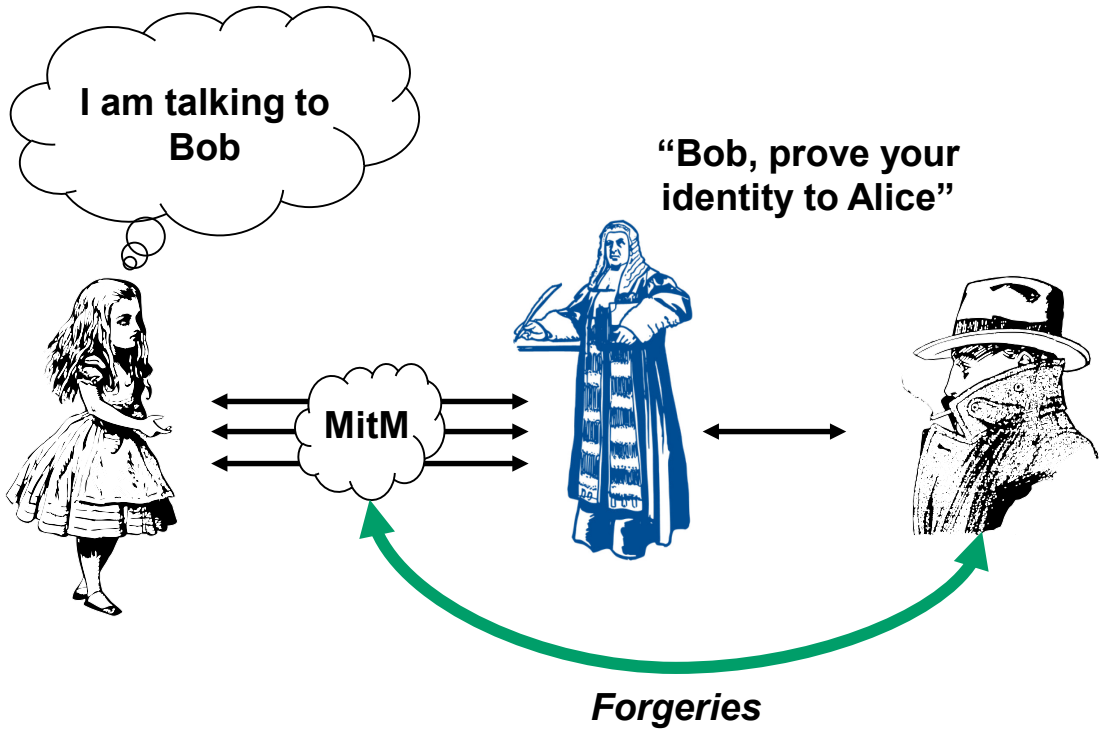
- Verifier becomes the endpoint
- Bob used as identity-key oracle

Solution – Online-deniable protocols:

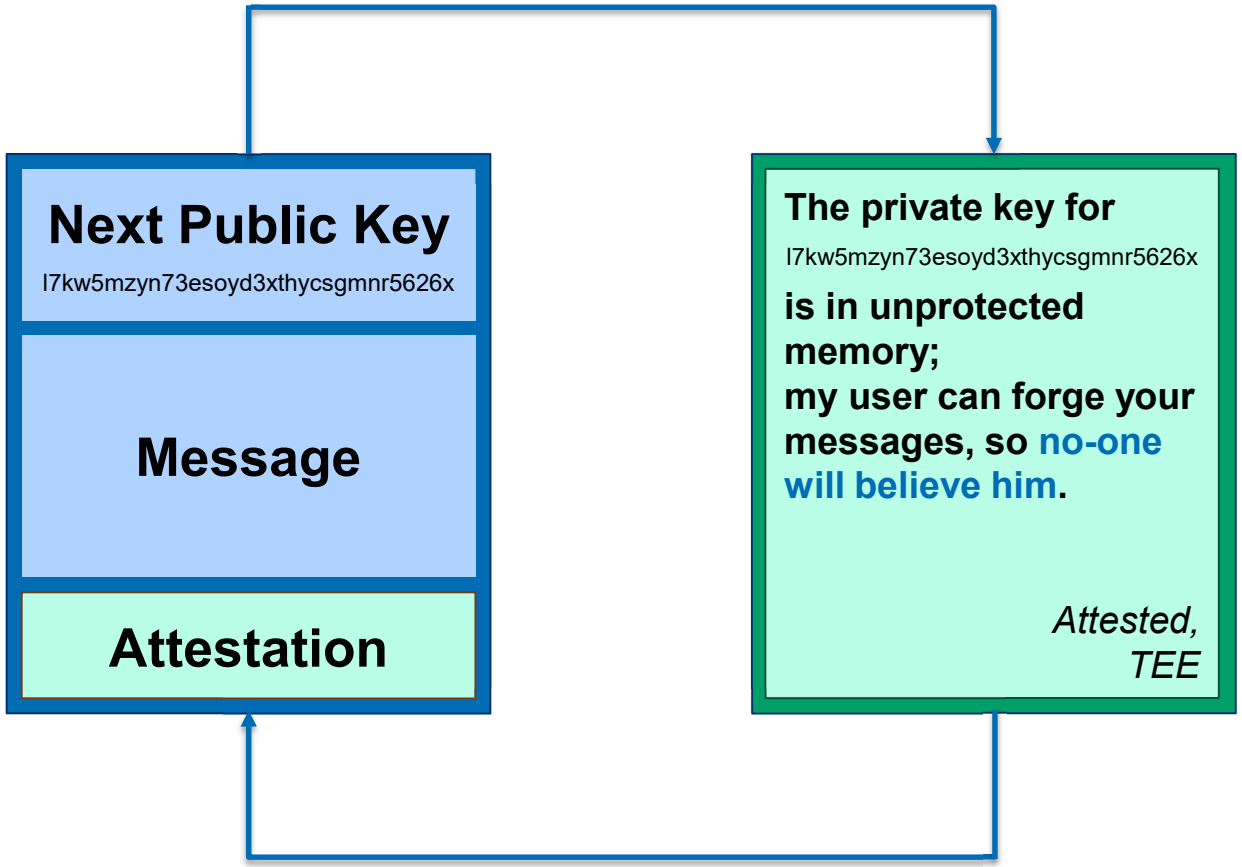
- Let identity-key holder MitM the session
- Verifier needs to trust Bob!

OTRv4 is online-deniable
https://github.com/otrv4/otrv4

Attack still possible if identity-key created within the attack TEE

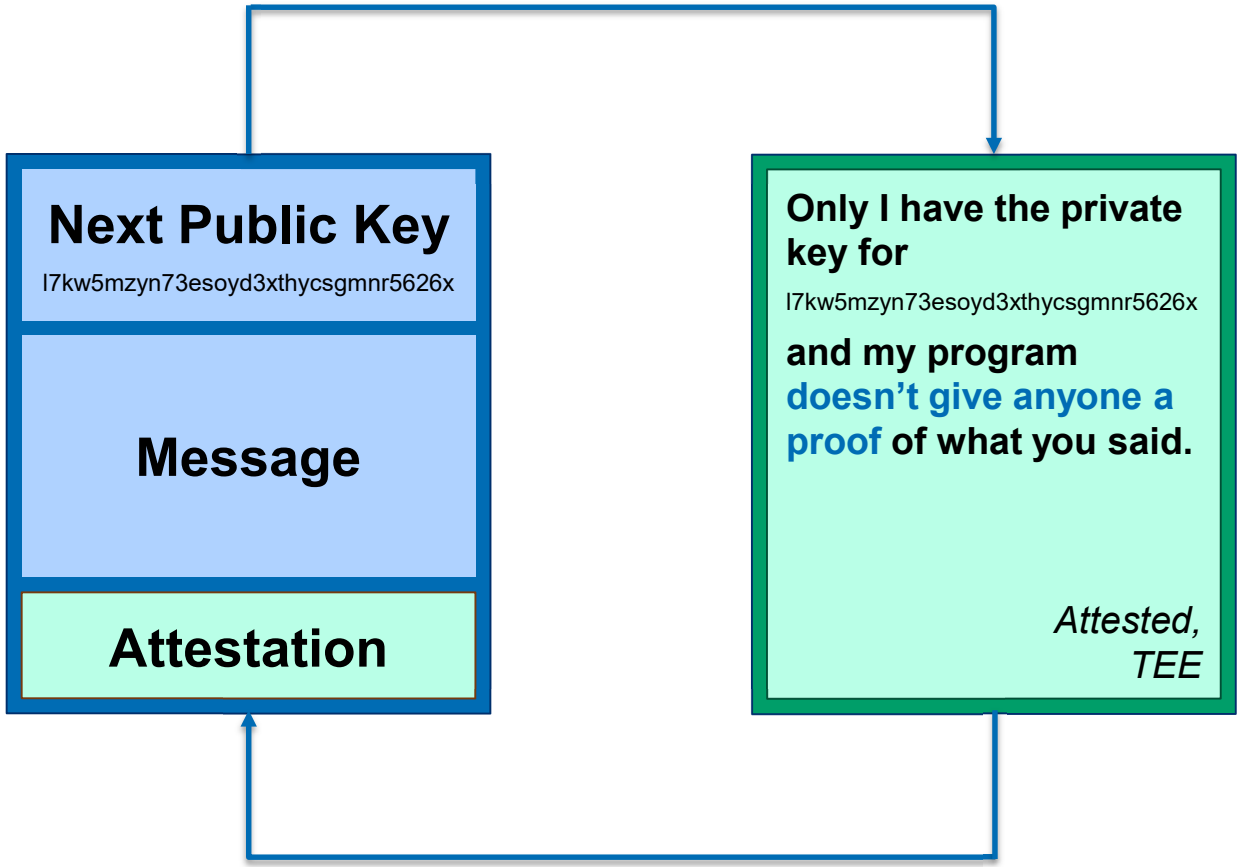


Defensive remote attestation



Use attestation to assure Alice about the behavior of Bob's TEE

Defensive remote attestation



Use attestation to assure Alice about the behavior of Bob's TEE

Put the human in the loop

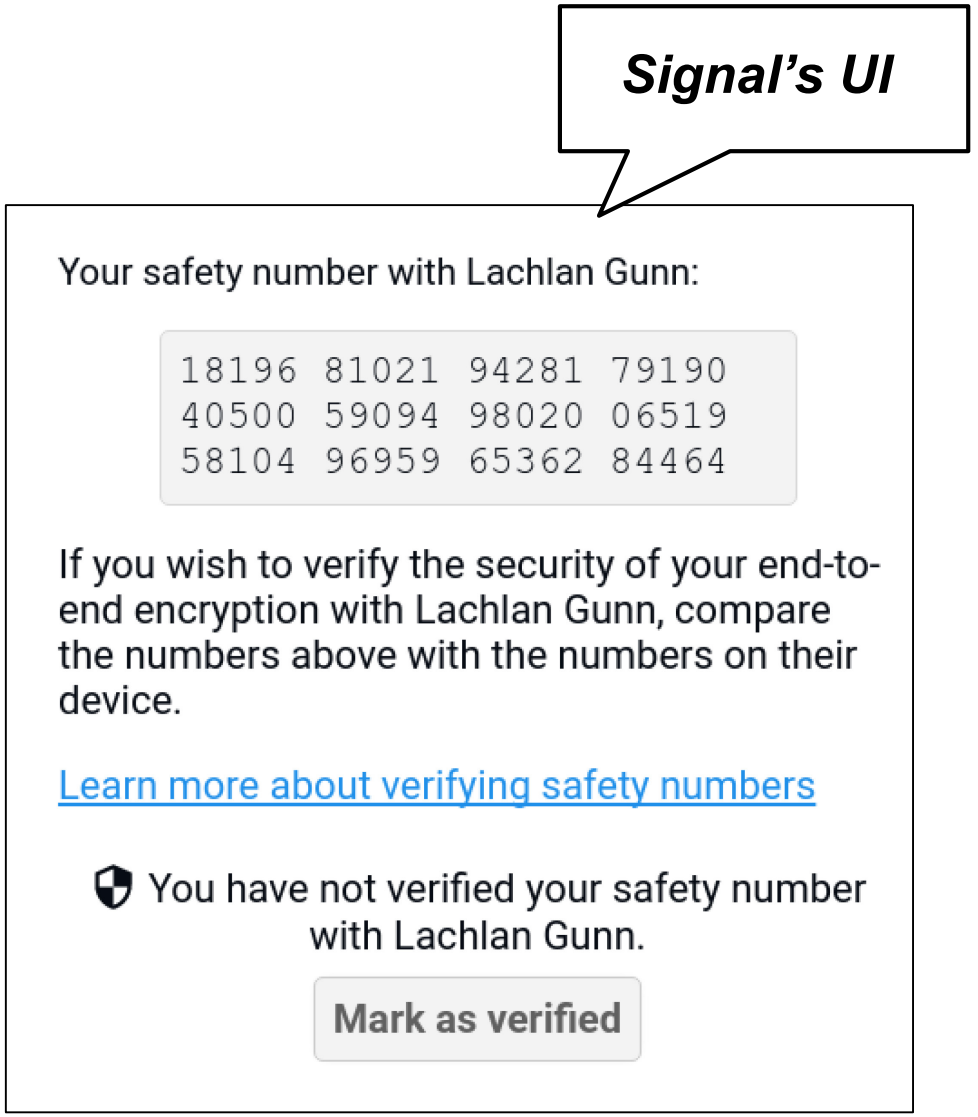
Hardware can only attest what is verifiable on the machine

Requiring human input is helpful:

- Use a different identity key for each recipient
- Verify fingerprints manually

Attack attests only a key but not who owns it

Compatible with current UIs



Countermeasures

Switch to online-deniable protocols

- *Deployability: high*
- *Effectiveness: medium*

Our recommendation

Defensive remote attestation

- *Deployability: low*
- *Effectiveness: high*

Put a human in the loop

- *Deployability: medium*
- *Effectiveness: medium*

The Big Picture

Not just messaging protocols

Anything machine-verifiable is at risk

TLS servers:

- Deniability of web-based messaging
- Proof of malware distribution

End-to-end verifiable voting

Black Hat Sound Bytes

- Deniability is important
- Attestation undermines deniability guarantees in messaging protocols
- Online-deniable protocols (e.g. OTRv4) reduce attack window

ia.cr/2018/424