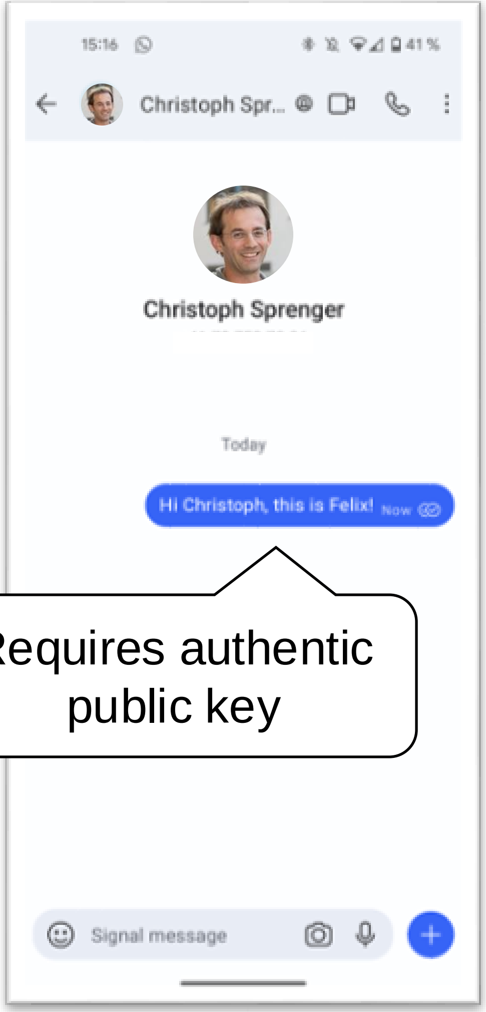
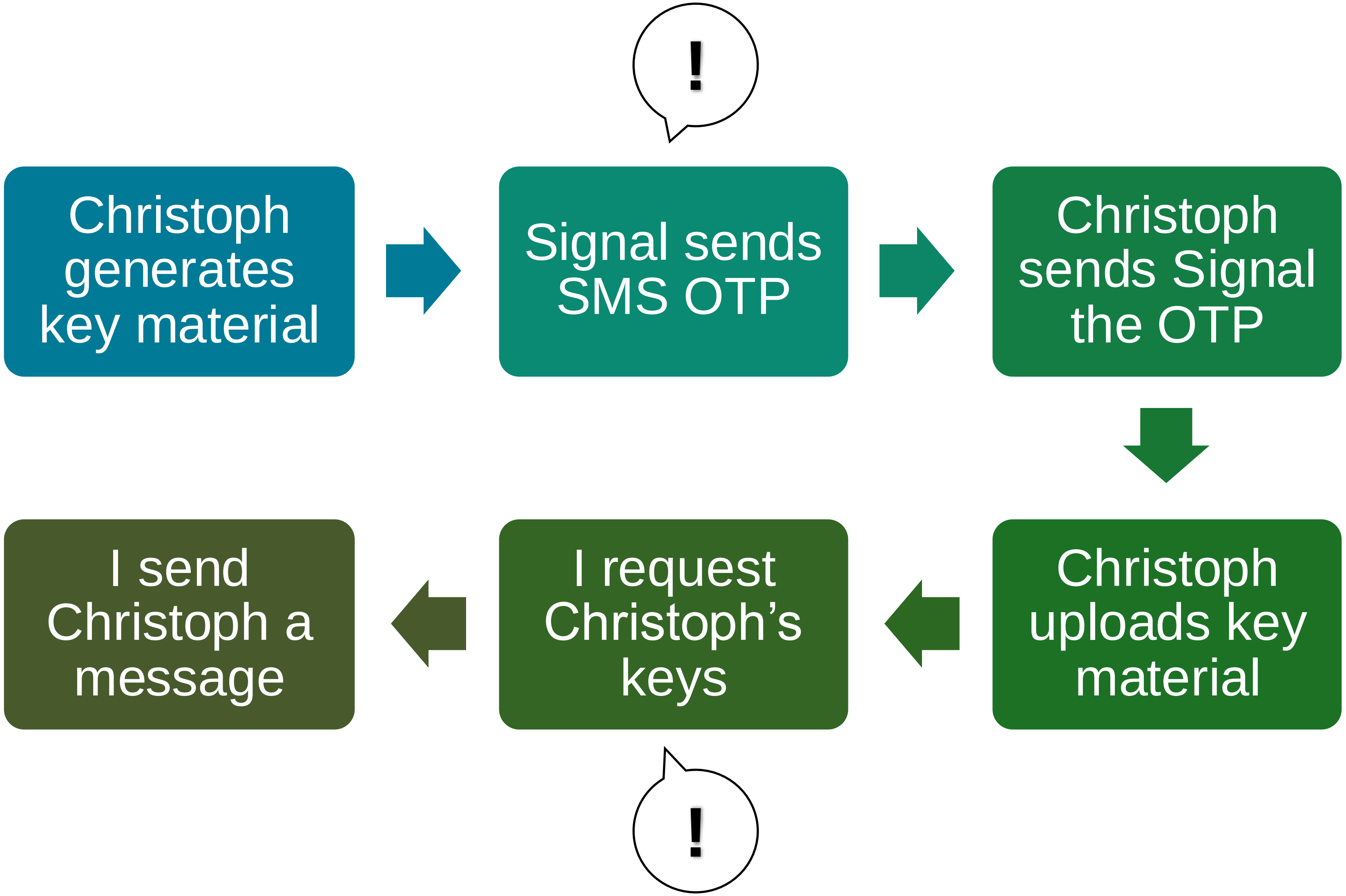
SOAP: A Social Authentication Protocol

Felix Linker, David Basin
Department of Computer Science

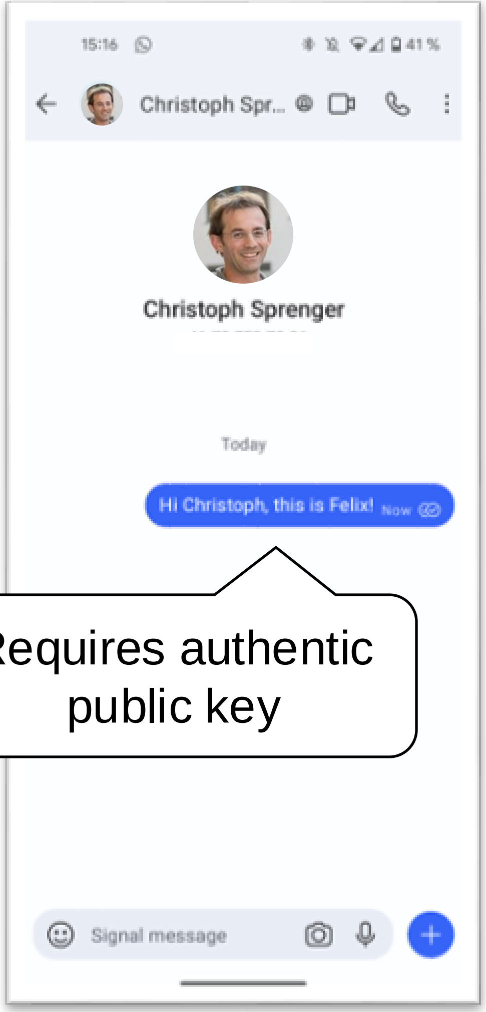
End-to-End Encrypted Messaging



Requires authentic public key



End-to-End Encrypted Messaging



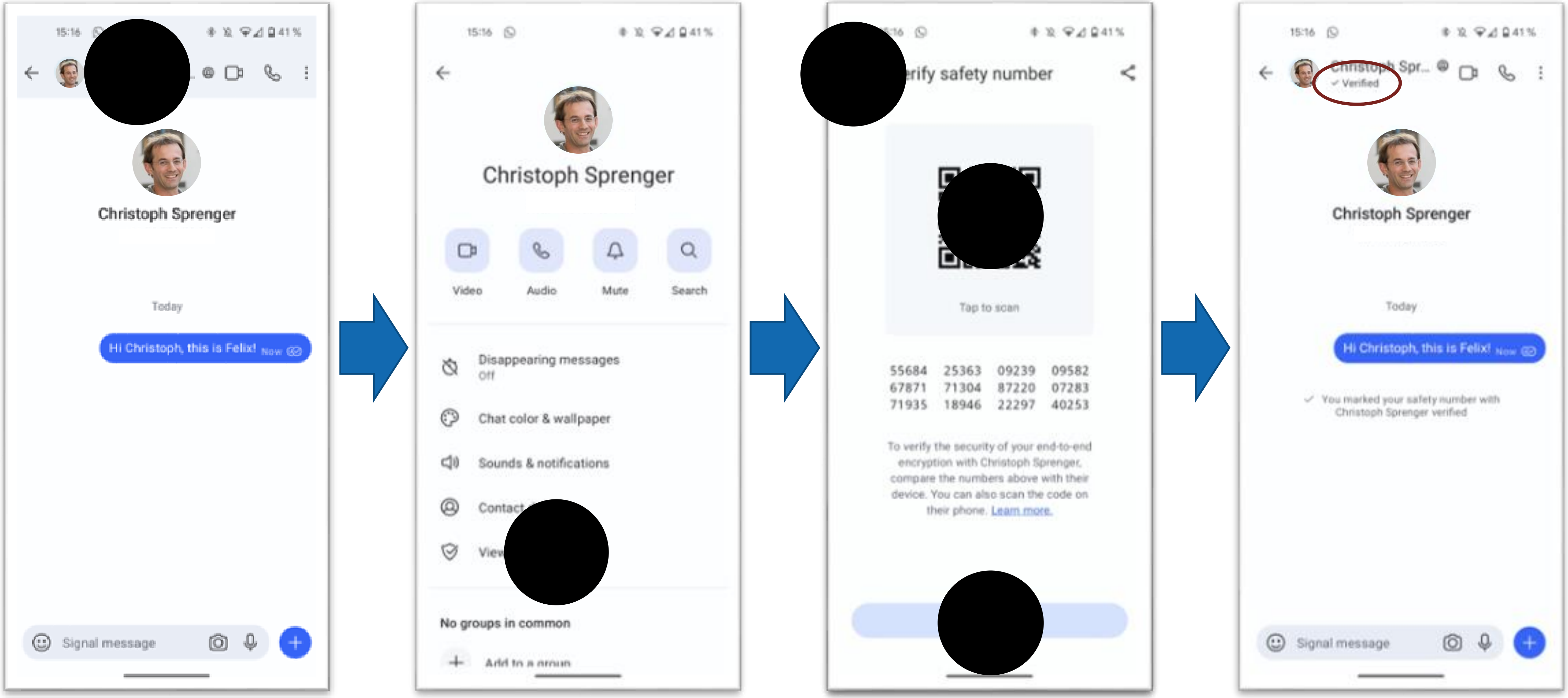
Requires authentic public key

Can we do better?

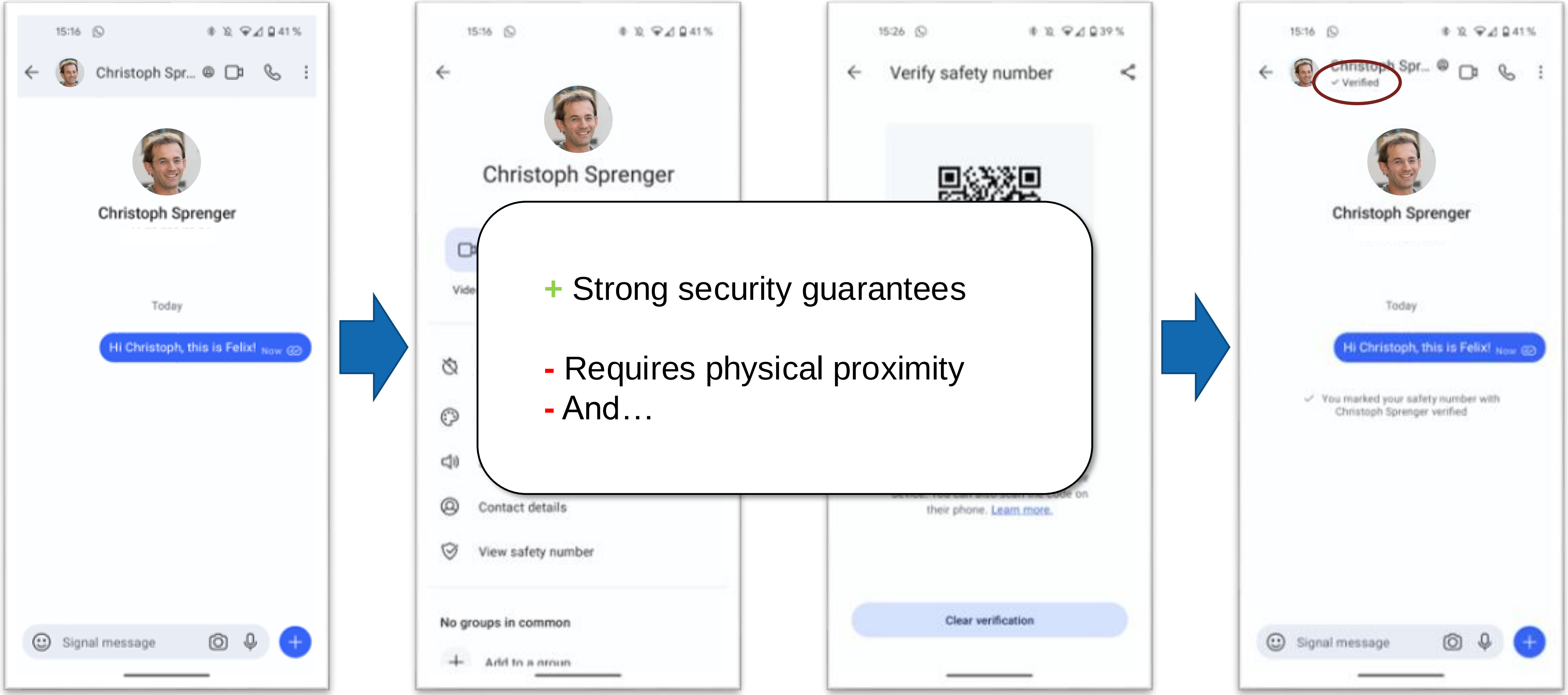
Christoph generates key material → Signal sends SMS OTP → Christoph sends Signal the OTP → Christoph uploads key material → I request Christoph's keys → I send Christoph a message

!

End-to-End Encrypted Messaging



End-to-End Encrypted Messaging

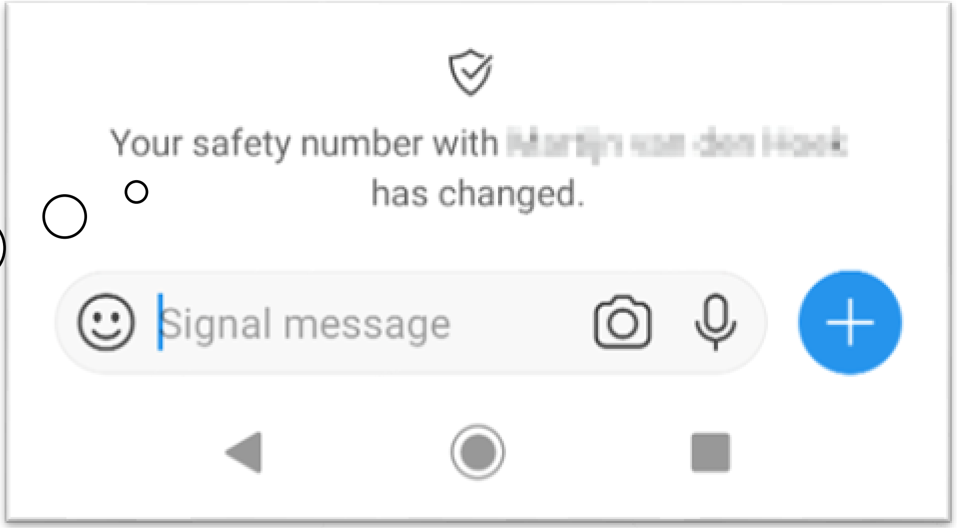


+ Strong security guarantees

- Requires physical proximity
- And…

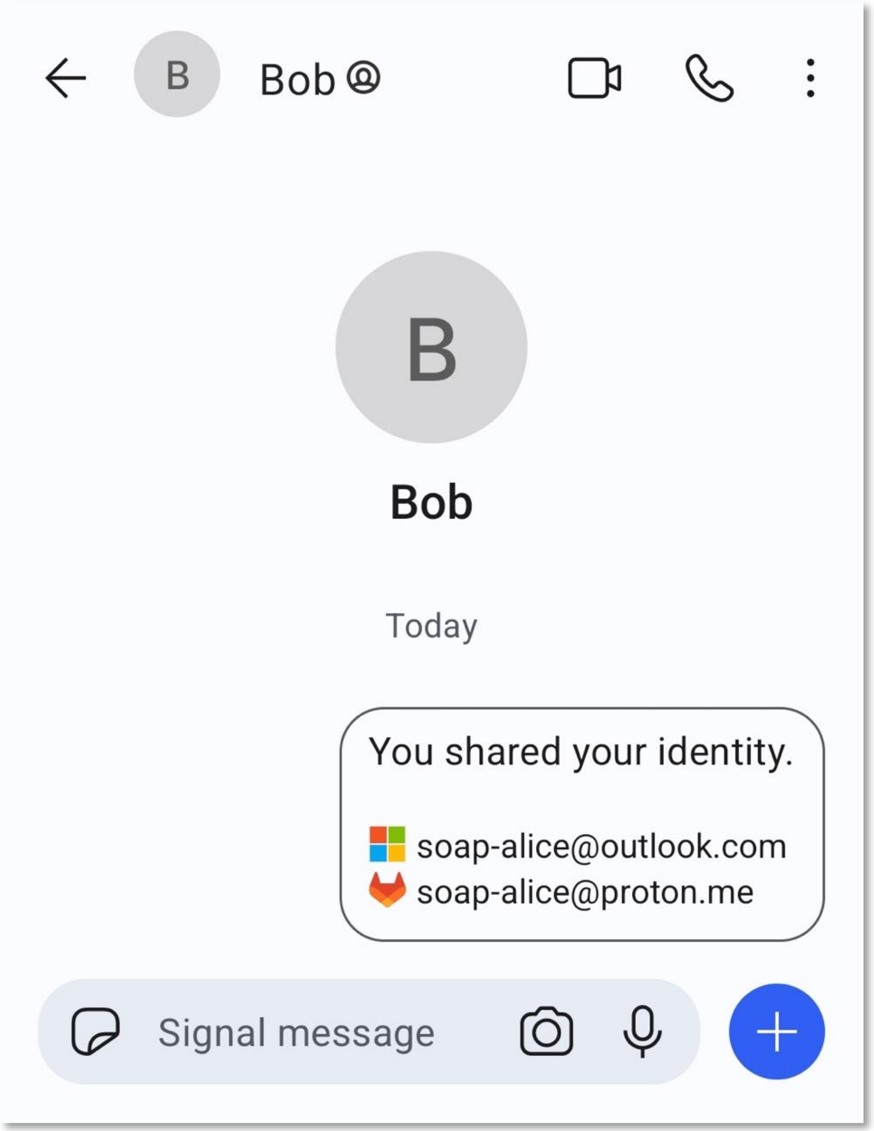
…what if there is an attack later?

Something happened and you need to scroll to get on with things…



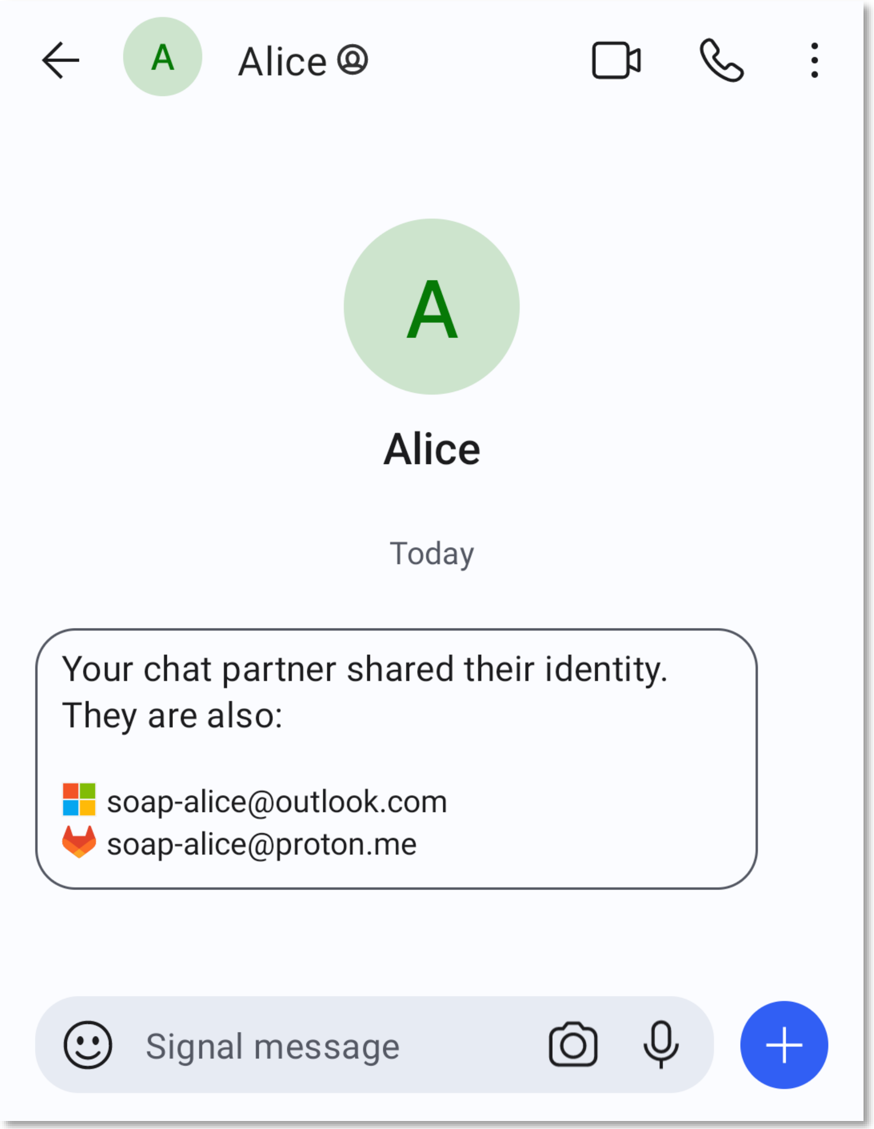
Can we do better?
Yes, use SOAP!

Alice to Bob

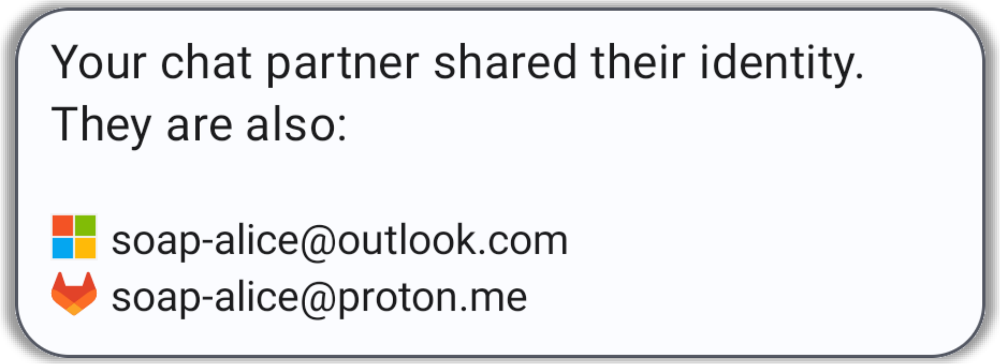


Alice runs SOAP for Bob

Bob to Alice

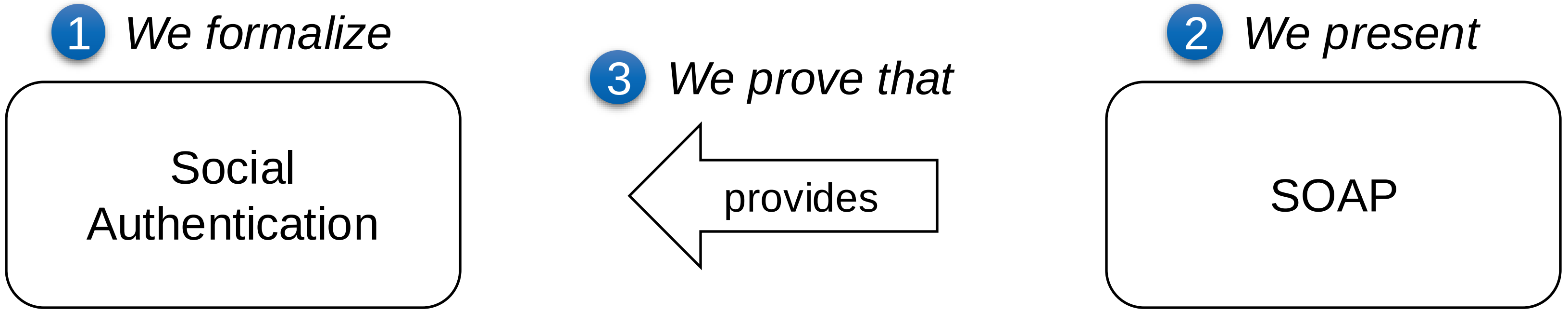


What can Bob conclude from this message?



- Bob **socially authenticates** Alice [Vaziripour et al., CHI 2019]
- The same person controlling the Signal account, controls given third-party accounts
- If Bob knows accounts, Bob can authenticate Alice
- If Bob doesn't know accounts, Bob can use them as second factor

Contributions

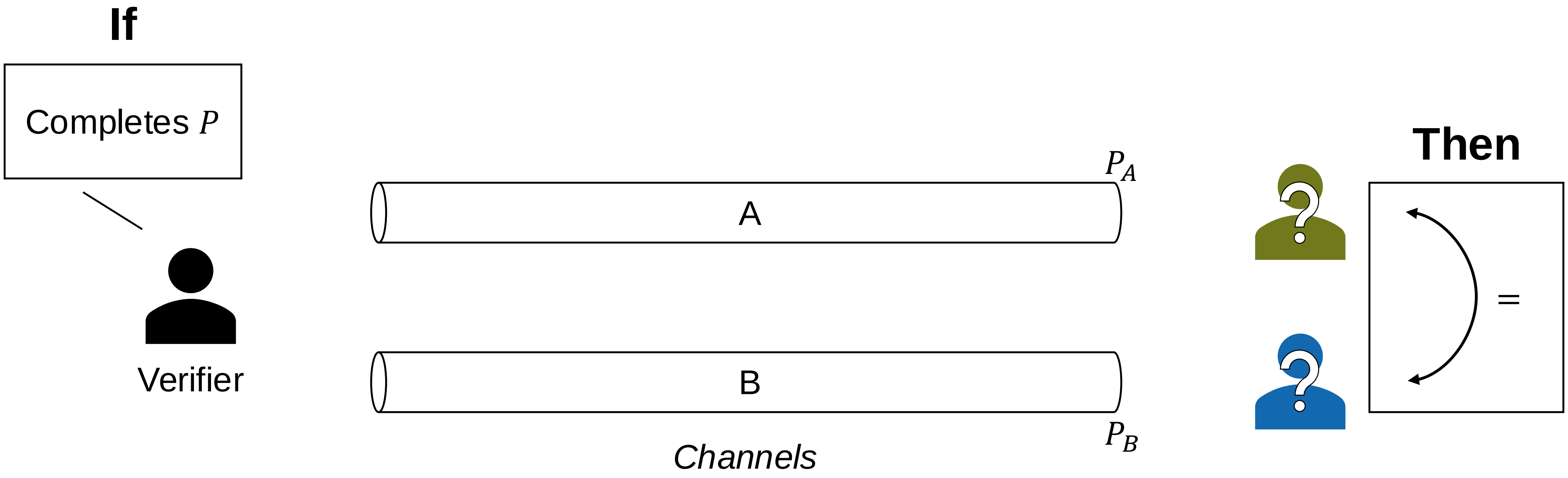


4 …and formally relate it to existing notions of authentication

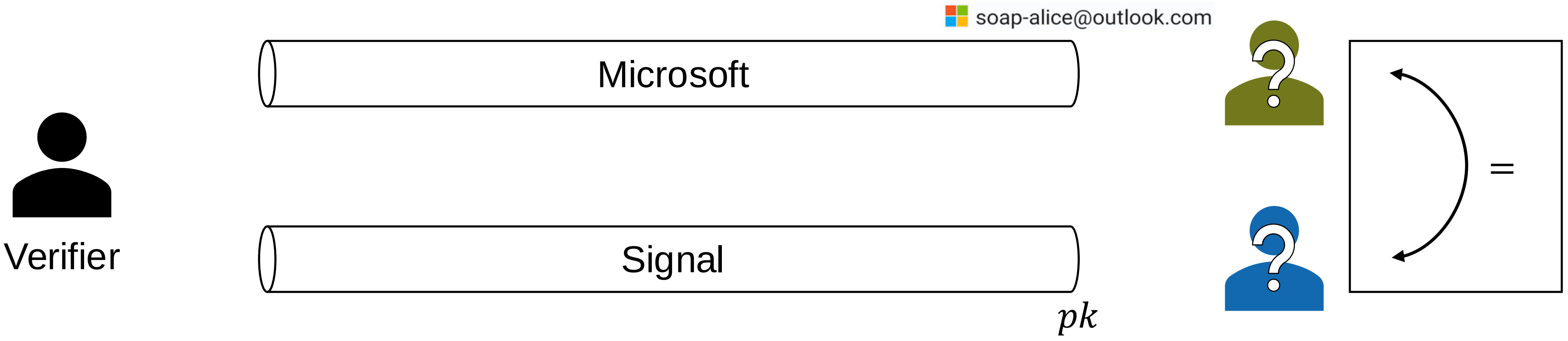
5 …and implement & evaluate prototypes

Social Authentication Formalized

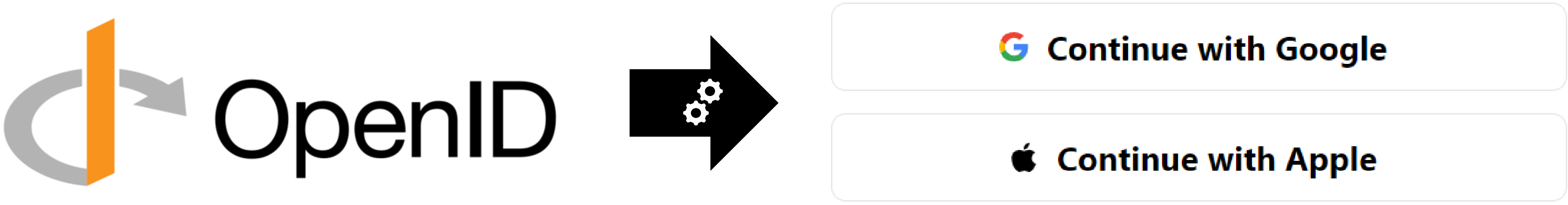
A Protocol P provides Social Authentication when…



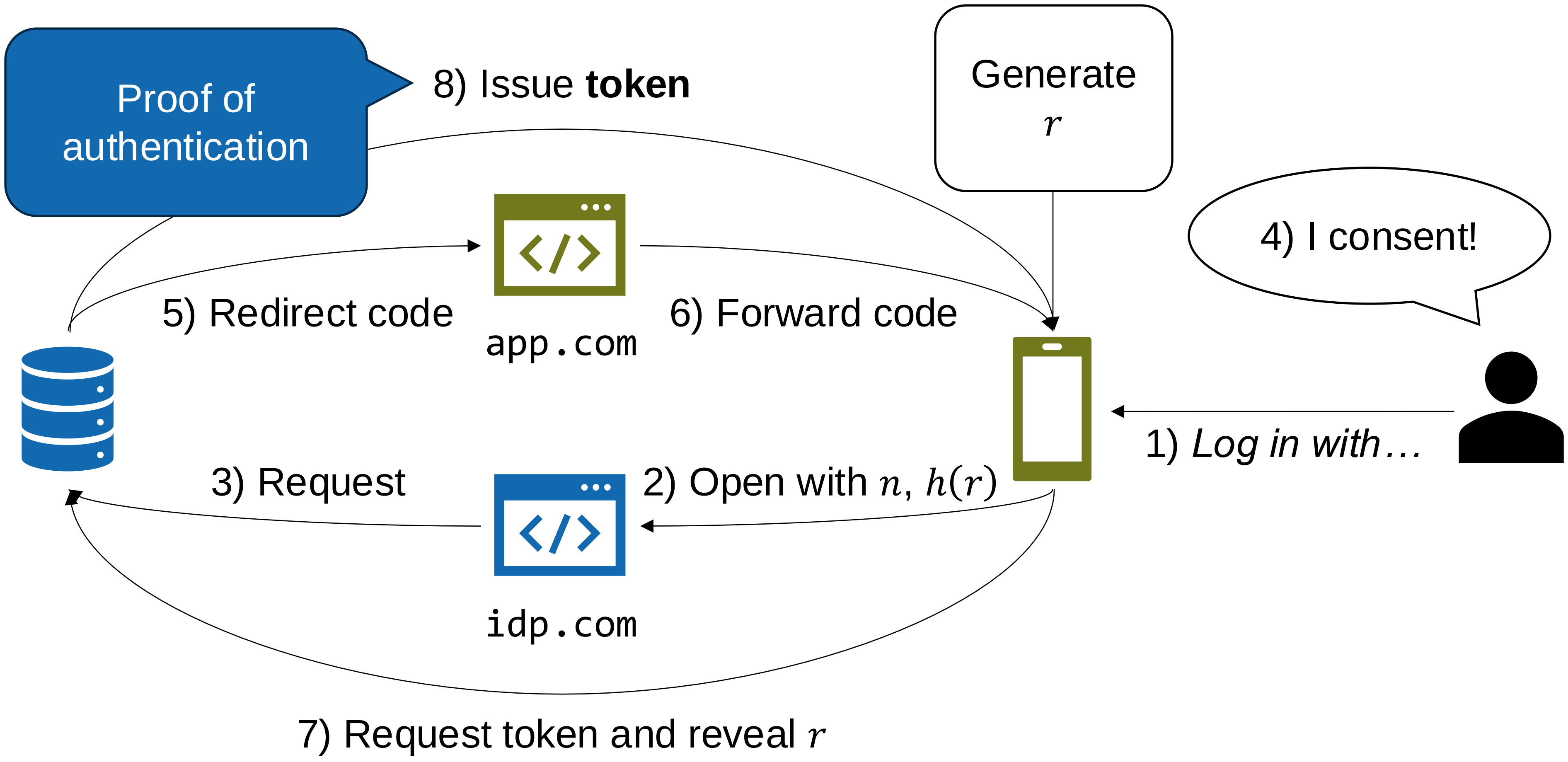
Social Authentication



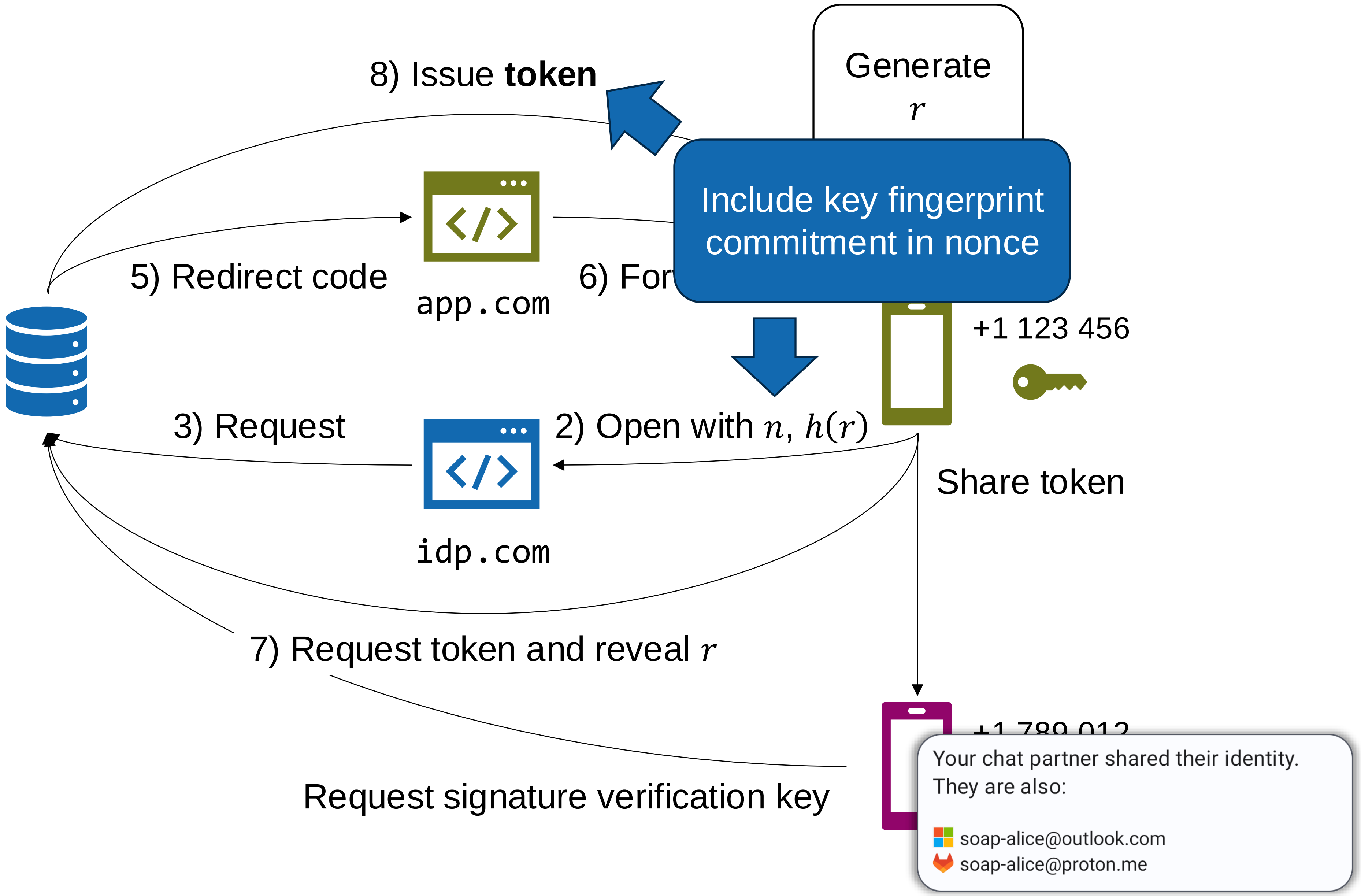
Automate Social Authentication using OpenID Connect



Automate Social Authentication using OpenID Connect

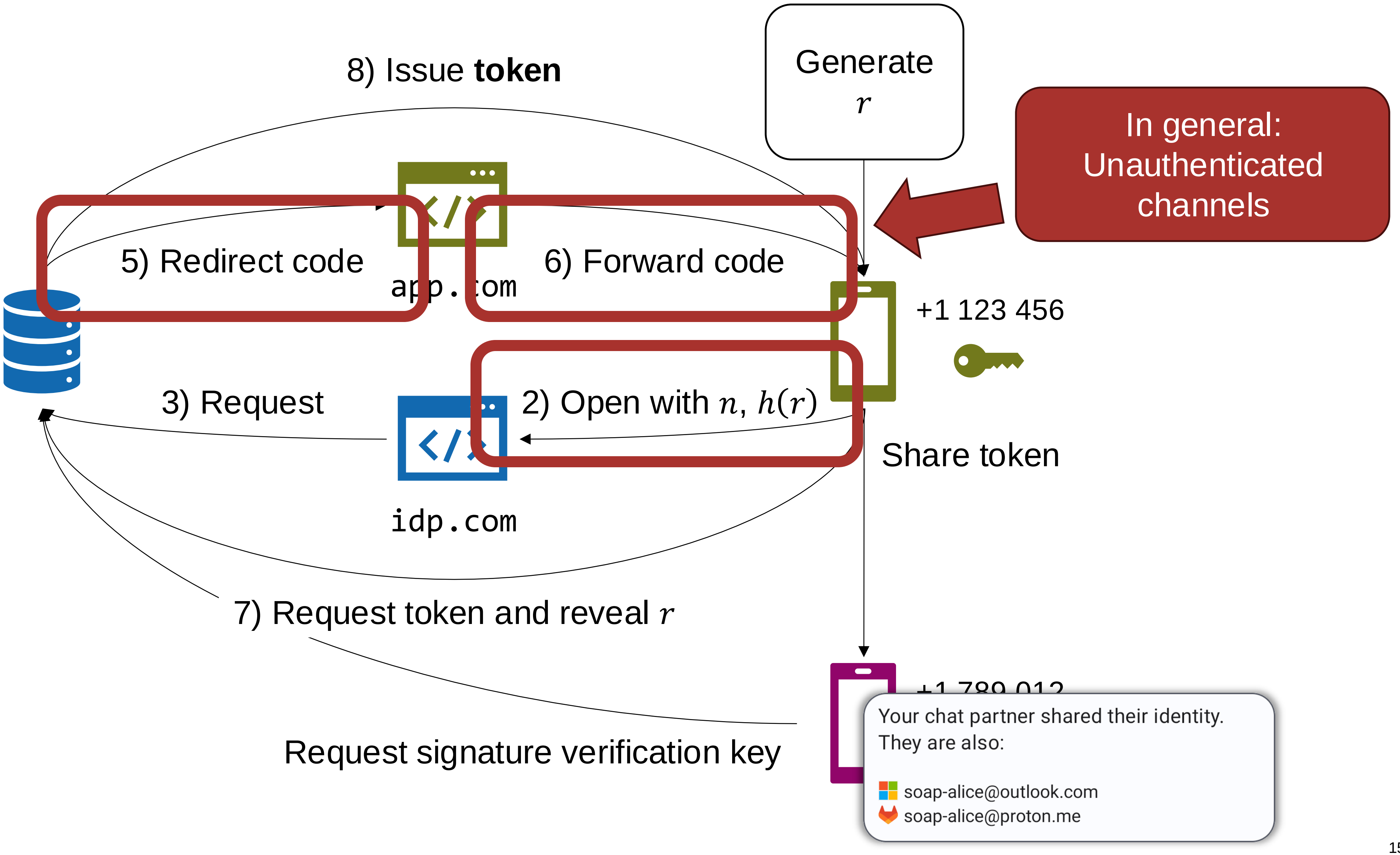


SOAP

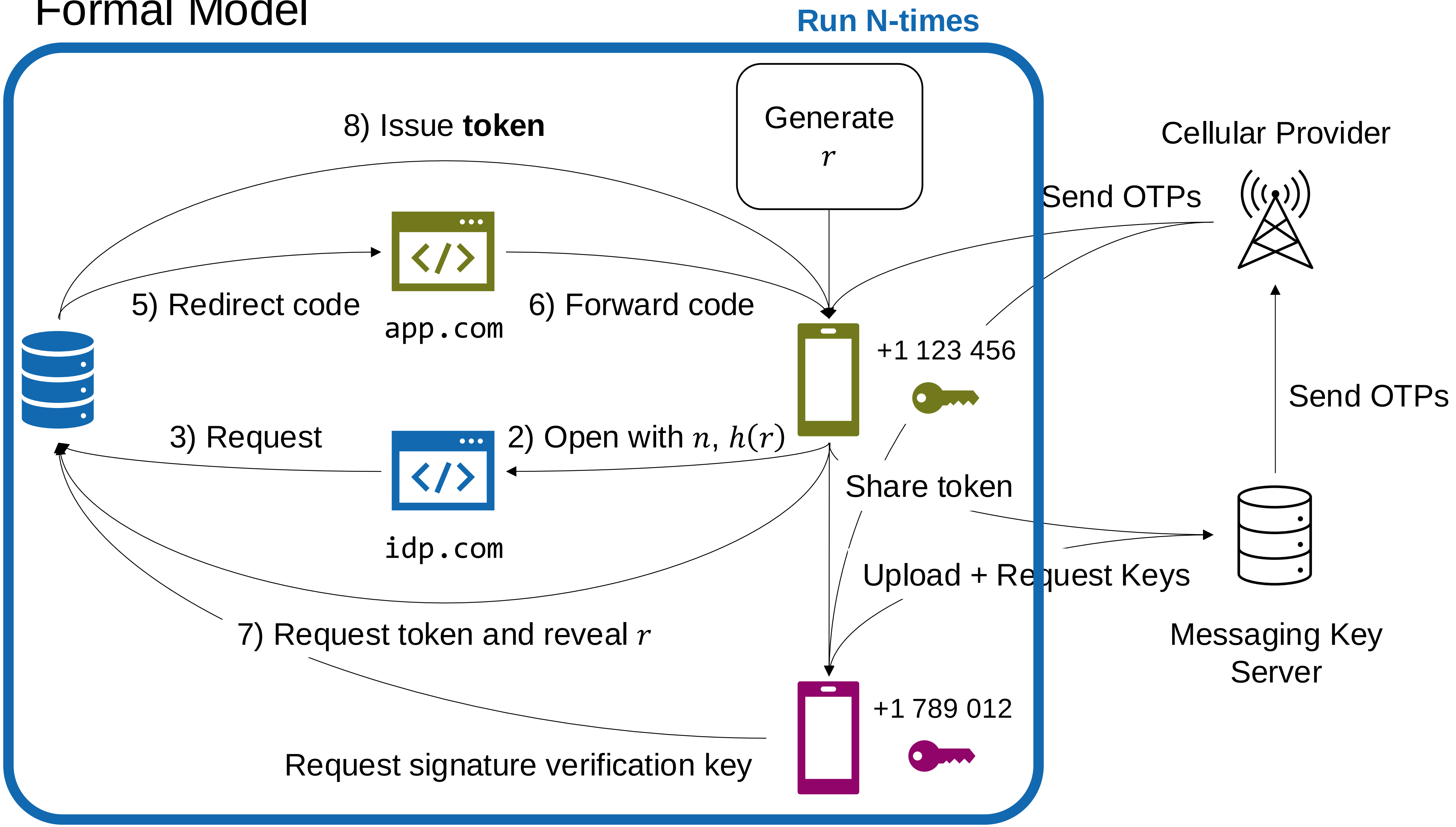


8) Issue **token**
Generate r
Include key fingerprint commitment in nonce
5) Redirect code
app.com
6) For
+1 123 456
3) Request
2) Open with n, $h(r)$
idp.com
Share token
7) Request token and reveal r
Request signature verification key
Your chat partner shared their identity.
They are also:
soap-alice@outlook.com
soap-alice@proton.me

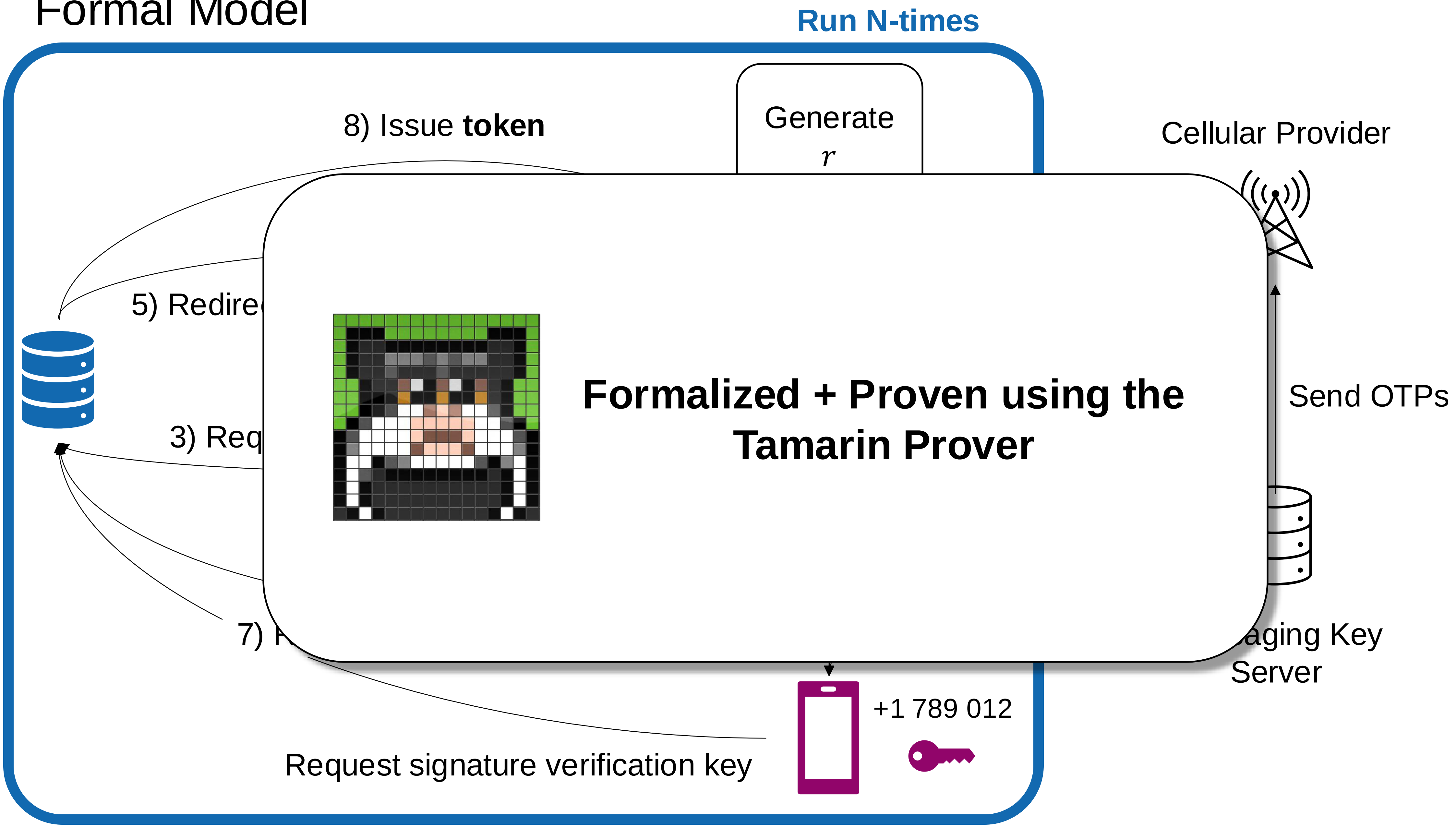
What could go wrong?



Formal Model



Formal Model



SOAP Security Guarantees

An adversary can intercept messages if…

Standard Security (They break SMS OTPs ∨ They break key server)

∧

+ SOAP They break <u>every</u> associated account

Account credentials uncompromised	*TLS keys uncompromised*	*Messaging application uncompromised*
Web browser uncompromised	*HTTP redirects remain confidential*	*Symbolic Model*

…assuming

Secure user behavior

…<u>not</u> assuming

SOAP Security Guarantees

An adversary can intercept messages if…

Standard Security (They break SMS OTPs ∨ They break key server)

∧

+ SOAP They break <u>every</u> associated account

In particular:
- Users click on all links
- Users consent to everything

Account credentials uncompromised

T unc

Trade-off!

ssaging application uncompromised

Web browser uncompromised

HTTP redirects remain confidential

Symbolic Model

Secure user behavior

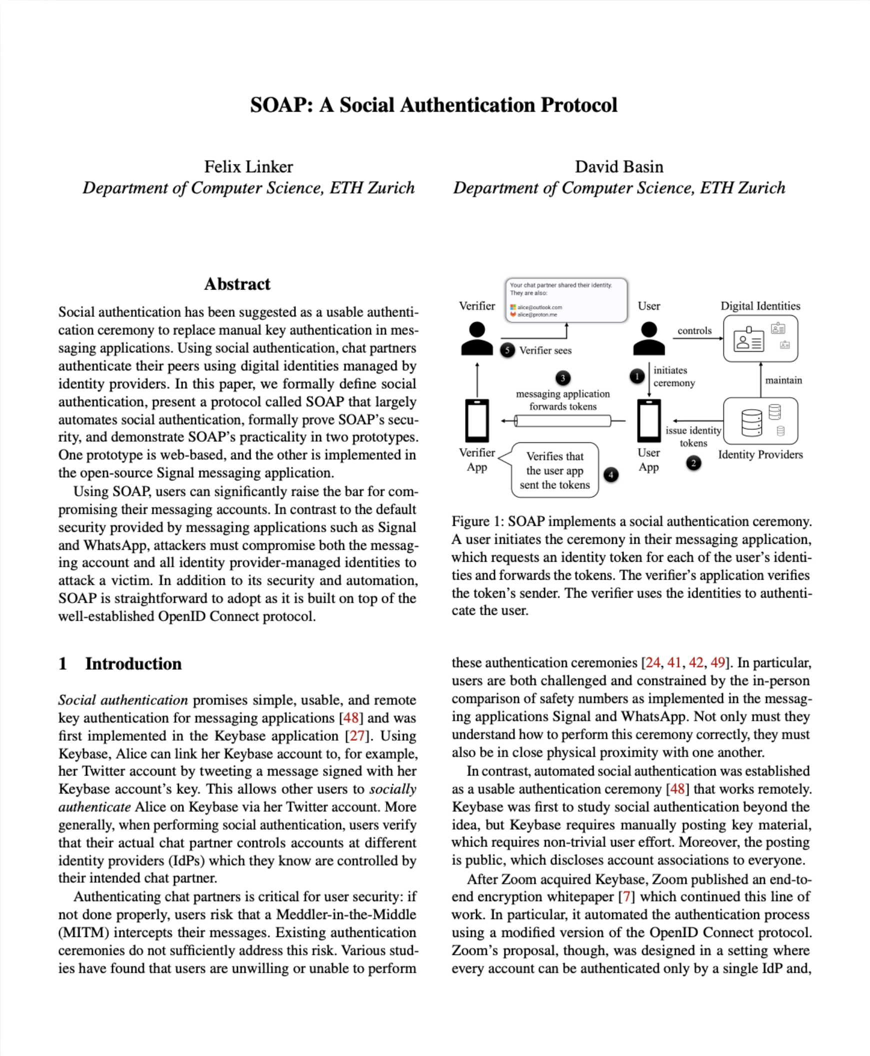
…assuming

…<u>not</u> assuming

Conclusion

- More on social authentication formally
- Protocol details
- More discussion on HTTP redirect assumption
- Comparison to other mechanisms

- Going forward
  - More OpenID Connect providers
  - Standardization
  - Make it work in the background
  - Finding the ideal UI
  - Combination with transparency logs
- Contributions presented
  - Social authentication formalized
  - SOAP: Automated Social Authentication
  - Proven to be secure



SOAP: A Social Authentication Protocol

Felix Linker
Department of Computer Science, ETH Zurich

David Basin
Department of Computer Science, ETH Zurich

Abstract

Social authentication has been suggested as a usable authentication ceremony to replace manual key authentication in messaging applications. Using social authentication, chat partners authenticate their peers using digital identities managed by identity providers. In this paper, we formally define social authentication, present a protocol called SOAP that largely automates social authentication, formally prove SOAP's security, and demonstrate SOAP's practicality in two prototypes. One prototype is web-based, and the other is implemented in the open-source Signal messaging application.

Using SOAP, users can significantly raise the bar for compromising their messaging accounts. In contrast to the default security provided by messaging applications such as Signal and WhatsApp, attackers must compromise both the messaging account and all identity provider-managed identities to attack a victim. In addition to its security and automation, SOAP is straightforward to adopt as it is built on top of the well-established OpenID Connect protocol.

Figure 1: SOAP implements a social authentication ceremony. A user initiates the ceremony in their messaging application, which requests an identity token for each of the user's identities and forwards the tokens. The verifier's application verifies the token's sender. The verifier uses the identities to authenticate the user.

1 Introduction

Social authentication promises simple, usable, and remote key authentication for messaging applications [48] and was first implemented in the Keybase application [27]. Using Keybase, Alice can link her Keybase account to, for example, her Twitter account by tweeting a message signed with her Keybase account's key. This allows other users to *socially authenticate* Alice on Keybase via her Twitter account. More generally, when performing social authentication, users verify that their actual chat partner controls accounts at different identity providers (IdPs) which they know are controlled by their intended chat partner.

Authenticating chat partners is critical for user security: if not done properly, users risk that a Meddler-in-the-Middle (MITM) intercepts their messages. Existing authentication ceremonies do not sufficiently address this risk. Various studies have found that users are unwilling or unable to perform these authentication ceremonies [24, 41, 42, 49]. In particular, users are both challenged and constrained by the in-person comparison of safety numbers as implemented in the messaging applications Signal and WhatsApp. Not only must they understand how to perform this ceremony correctly, they must also be in close physical proximity with one another.

In contrast, automated social authentication was established as a usable authentication ceremony [48] that works remotely. Keybase was first to study social authentication beyond the idea, but Keybase requires manually posting key material, which requires non-trivial user effort. Moreover, the posting is public, which discloses account associations to everyone.

After Zoom acquired Keybase, Zoom published an end-to-end encryption whitepaper [7] which continued this line of work. In particular, it automated the authentication process using a modified version of the OpenID Connect protocol. Zoom's proposal, though, was designed in a setting where every account can be authenticated only by a single IdP and,

@felixlinker

felixlinker.de