

# **CSE 410/565: Computer Security**

Instructor: Dr. Ziming Zhao

# Last Class

- User-chosen secrets for authentication
  - Text-based password
  - Picture gesture authentication

# Entity Authentication

- Identification mechanisms are often divided into 3 types based on how the identity evidence is gathered
  - user knows a secret
    - examples include passwords, personal identification numbers (PINs), secret keys, mother's maiden name, etc.
  - user possesses a token
    - these are normally hardware tokens such as magnetic-striped cards or custom-designed devices for time-variant passwords
  - user has a physical attribute
    - characteristics inherent to the user such as biometrics, handwritten signatures, keystroke dynamics, facial and hand geometries, voice, etc.

# Remote Authentication

- Now assume we want to use passwords for **remote authentication**
  - will it work?
- Passwords observed on the network are trivially susceptible to replay
  - initially remote login and file transfer programs, such as `telnet`, communicated passwords in the clear
  - now encryption is used (`ssh`, `scp`, etc.)
- Authentication based on **time-invariant passwords** is therefore a **weak form of authentication**
  - this form of authentication is nevertheless the most common
- A natural way to improve security is to use **one-time passwords**

# One-Time Passwords

- In authentication based on **one-time passwords** each password is used only once
- Such authentication can be realized in the following ways:
  - the user and the system initially **agree on a sequence of passwords**
    - simple solution but requires maintenance of the shared list
  - the **user updates her password** with each instance of the authentication protocol
    - e.g., the user might send the new password encrypted under a key derived from the current password
    - this method crucially relies on the correct communication of the new password to the system

# One-Time Passwords

- Leslie Lamport, 1984

2013 Turing Award: For fundamental contributions to the theory and practice of distributed and concurrent systems, notably the invention of concepts such as causality and logical clocks, safety and liveness, replicated state machines, and sequential consistency.

Technical Note  
Operating Systems

Anita K. Jones  
Editor

## Password Authentication with Insecure Communication

Leslie Lamport  
SRI International

A method of user password authentication is described which is secure even if an intruder can read the system's data, and can tamper with or eavesdrop on the communication between the user and the system. The method assumes a secure one-way encryption function and can be implemented with a microcomputer in the user's terminal.

Key Words and Phrases: security, authentication, passwords, one-way function

CR Categories: 4.35, 4.39

### I. The Problem

In remotely accessed computer systems, a user identifies himself to the system by sending a secret password. There are three ways an intruder could learn the user's secret password and then impersonate him when interacting with the system:

- (1) By gaining access to the information stored inside the system, e.g., reading the system's password file.
- (2) By intercepting the user's communication with the system, e.g., eavesdropping on the line connecting the user's terminal with the system, or observing the execution of the password checking program.
- (3) By the user's inadvertent disclosure of his password,

ample, a voice print. Such a mechanism is beyond the scope of this paper, so we restrict ourselves to the problem of removing the first two weaknesses.

### II. The Solution

The first weakness can be eliminated by using a *one-way function* to encode the password. A one-way function is a mapping  $F$  from some set of words into itself such that:

- (1) Given a word  $x$ , it is easy to compute  $F(x)$ .
- (2) Given a word  $y$ , it is not feasible to compute a word  $x$  such that  $y = F(x)$ .

We will not bother to specify precisely what "easy" and "feasible" mean, so our reasoning will be informal. Note that given  $F(x)$ , it is always possible to find  $x$  by an exhaustive search. We require that such a computation be too costly to be practical. A one-way function  $F$  can be constructed from a secure encryption algorithm: one computes  $F(x)$  by encrypting a standard word using  $x$  as a key [1].

Instead of storing the user's password  $x$ , the system stores only the value  $y = F(x)$ . The user identifies himself by sending  $x$  to the system; the system authenticates his identity by computing  $F(x)$  and checking that it equals the stored value  $y$ . Authentication is easy, since our first assumption about  $F$  is that it is easy to compute  $F(x)$  from  $x$ . Anyone examining the system's permanently stored information can discover only  $y$ , and by the second assumption about  $F$  it will be infeasible for him to compute a value  $x$  such that  $y = F(x)$ . This is a widely used scheme, and is described in [2] and [3].

While removing the first weakness, this method does not eliminate the second—an eavesdropper can discover the password  $x$  and subsequently impersonate the user. To prevent this, one must use a sequence of passwords  $x_1, x_2, \dots, x_{1000}$ , where  $x_i$  is the password by which the user identifies himself for the  $i$ th time. (Of course, the value 1000 is quite arbitrary. The assumption we will tacitly make is that 1000 is small enough so that it is

# One-Time Passwords

- One-time password authentication mechanisms (cont.)
  - the new password is derived with each instance of the authentication protocol using a one-way hash function
    - the system based on hash chains is called S/Key and is due to Lamport
    - a user begins with secret  $k$  and produces a sequence of values  $k, h(k), h(h(k)), \dots, h^t(k)$
    - password for  $i$ th identification session is  $p_i = h^{t-i}(k)$
    - The server is given  $h^t(k)$  by the user
    - when user authenticates  $(i + 1)$ st time with  $p_{i+1}$ , the server checks whether  $h(p_{i+1}) = p_i$
    - if  $h$  is infeasible to invert, this convinces the server that the user is legitimate

# One-Time Passwords

- Example of S/Key

- suppose  $t = 5$
- at setup stage
  - user chooses  $k$  and computes  $h(k), h(h(k)), h^3(k), h^4(k), h^5(k)$
  - user gives  $h^5(k)$  to the verifier
- during authentication
  - at session 1:
  - at session 2:
  - at session 5:



# Entity Authentication

- An even **stronger form of authentication** is one where the user doesn't have to send the secret to the verifier
  - ideally you want to convince the verifier without leaking information about your secret
  - such solutions exist and often involve the verifier sending a random **challenge** to the claimant
  - the claimant uses the challenge and the secret to compute the **response**
  - anyone who monitors the channel, cannot deduce information about the secret

# Challenge-Response Techniques

- The goal of challenge-response techniques is to
  - use a single secret for authentication
  - provide evidence of the secret without leaking information about it
  - proving possession of a secret without leaking information about it is called a zero-knowledge proof of knowledge
- Challenge-response protocols can be built
  - from simple cryptographic primitives (e.g, MACs and signature schemes)
  - from scratch (Schnorr, Okamoto, and Guillou-Quisquater schemes)

# Challenge-Response Techniques

- The basic form of such protocols is normally as follows:
  - suppose Alice is authenticating to Bob
  - Alice has a secret  $s$  and Bob has a verification value  $v$
  - Bob sends to Alice a challenge  $c$  (chosen or computed anew)
  - Alice computes a response  $r = f(s, c)$  and sends it to Bob
  - Bob verifies  $r$  using  $c$  and  $v$
- Building a secure challenge-response protocol is non-trivial
  - must be secure against **active adversaries**
    - parallel session attack
    - man-in-the-middle attack

# Token-Based Authentication

- Authentication based on what you possess can be done using different types of tokens
  - memory cards
    - data is passively stored on a medium
    - a card reader can retrieve information stored on the card
    - e.g., magnetic stripe credit cards, ATM cards, hotel keys
    - memory cards provide a limited level of security (i.e., card contents can be read by any reader and copied to another card)
    - memory cards are often combined with a password or PIN
    - using memory cards with computers requires special reader

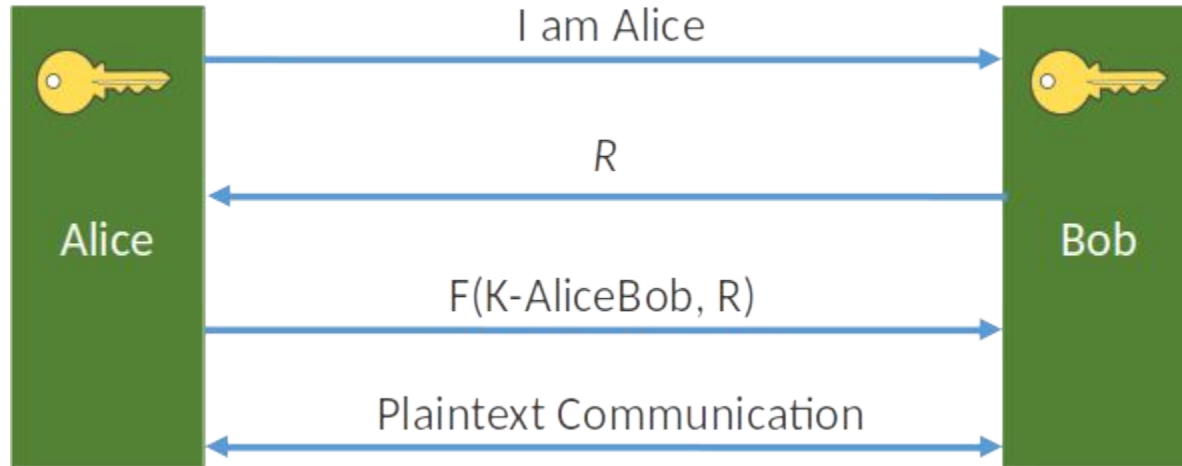
# Token-Based Authentication

- Types of authentication tokens (cont.)
  - smart cards
    - such cards have a built-in microprocessor, programmable read-only memory and random-access memory (RAM)
    - they can engage in different types of authentication protocols including challenge-response
    - such tokens can also be used to generate dynamic passwords
      - each minute the device generates a new password
      - the device and the verifier must be synchronized
    - tamper-resistance of such tokens must be addressed
      - it's been shown in the past that key material can be recovered with relatively inexpensive equipment

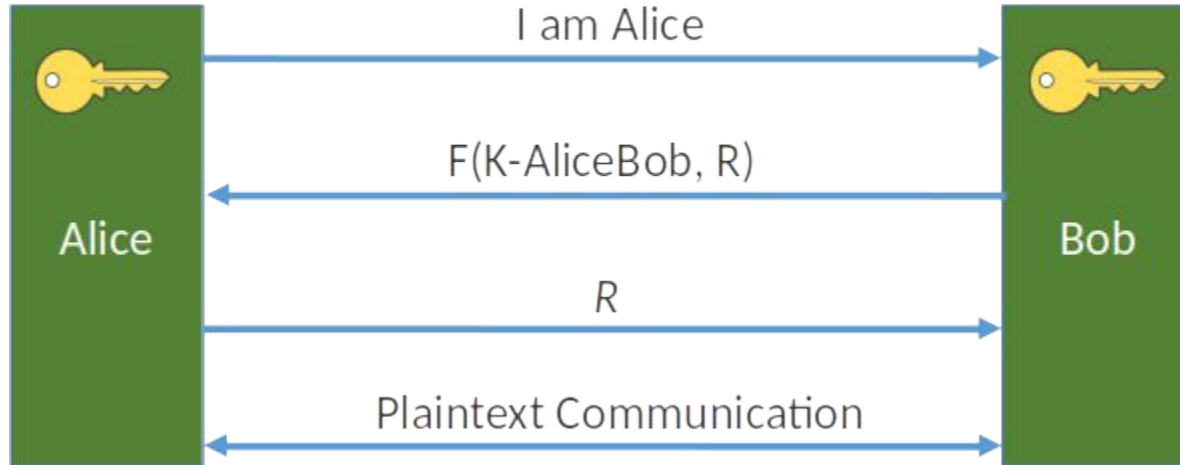
# Token-Based Authentication

- Types of authentication tokens (cont.)
  - USB dongle
    - USB tokens can also be used for authentication
    - they can store static data as well as code
      - recent dongles also include non-volatile memory
    - no additional hardware such a special-purpose reader is necessary
    - USB dongles are commonly used for copy protection of copyrighted material
    - dongle products often don't provide enough security to be used in rigid security requirement environments

# Authentication with Shared Secret

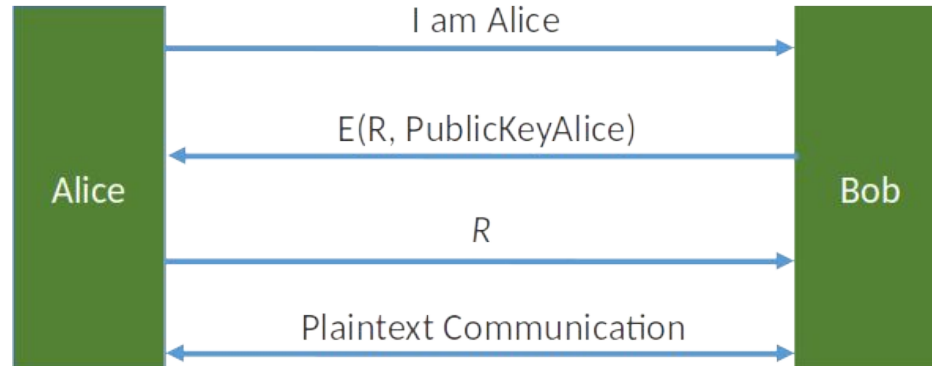
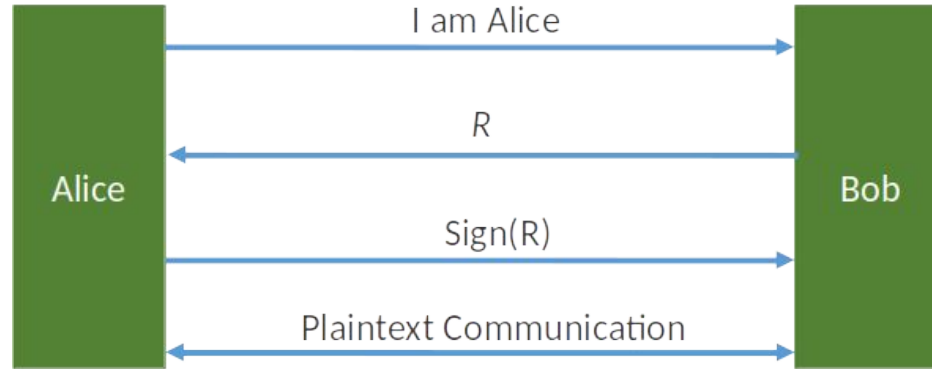


# Authentication with Shared Secret



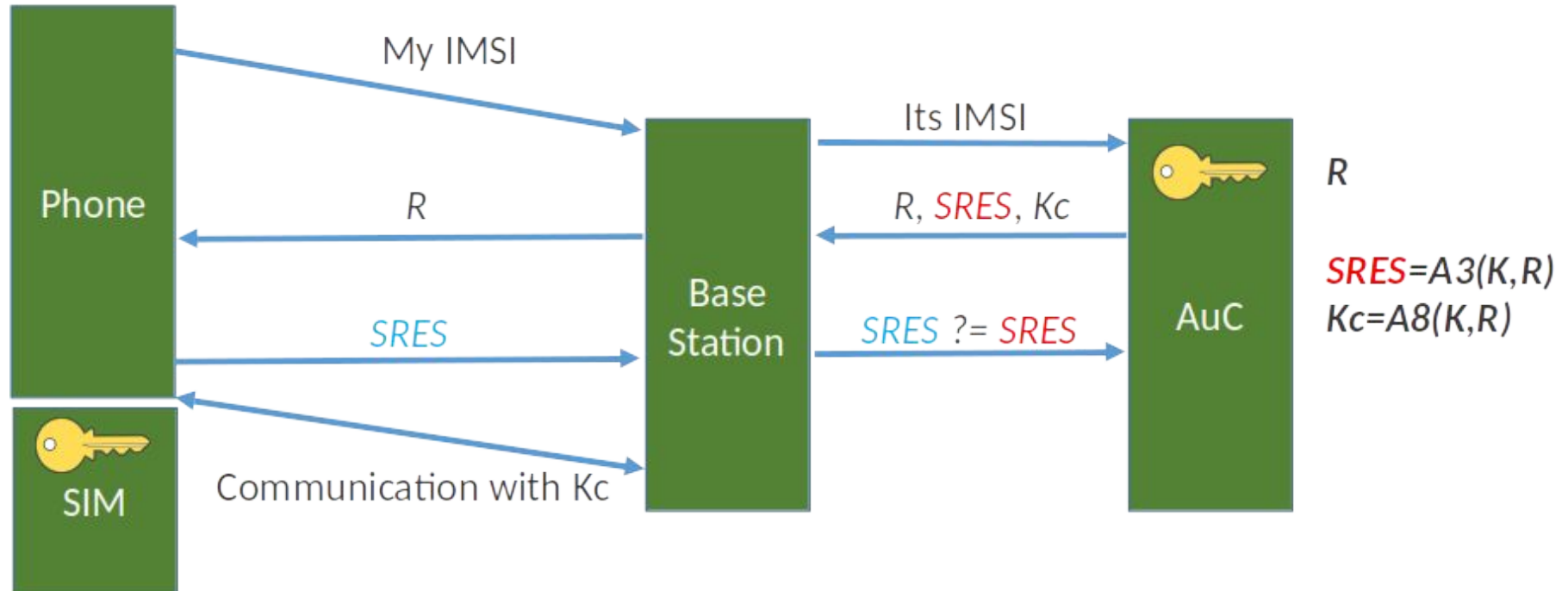


# Authentication with Public Key

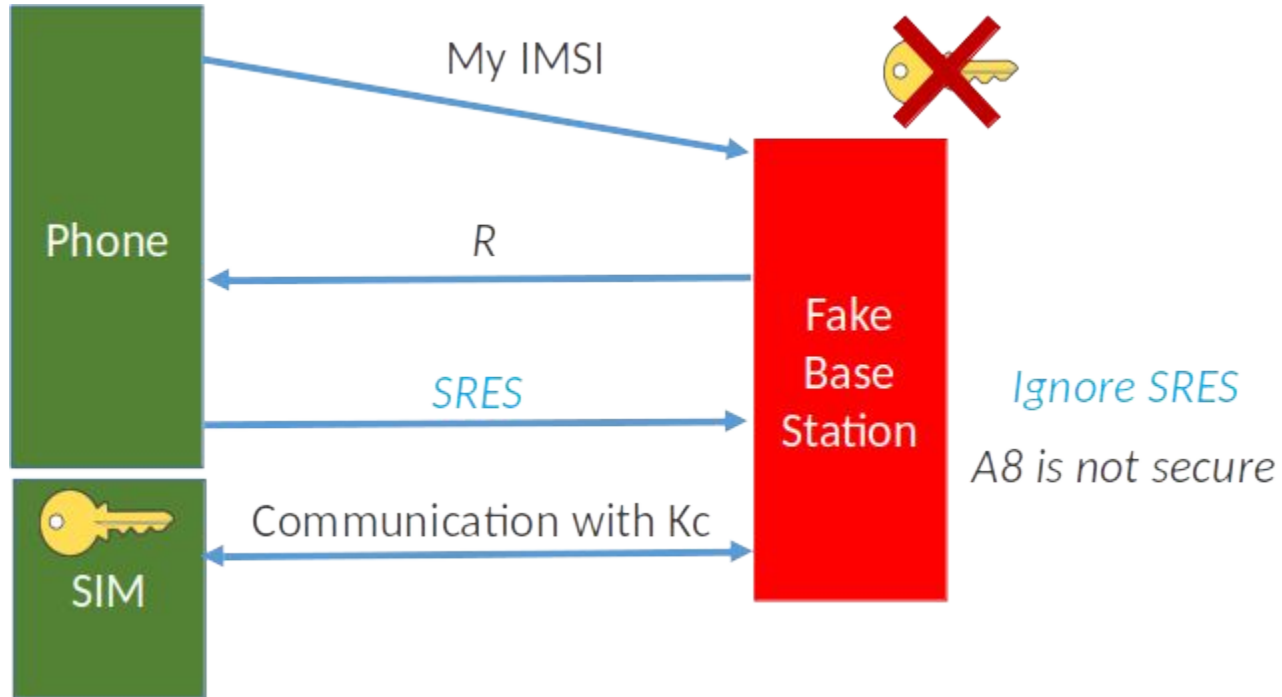


# GSM 2G Authentication

Carrier



# Fake Base Station

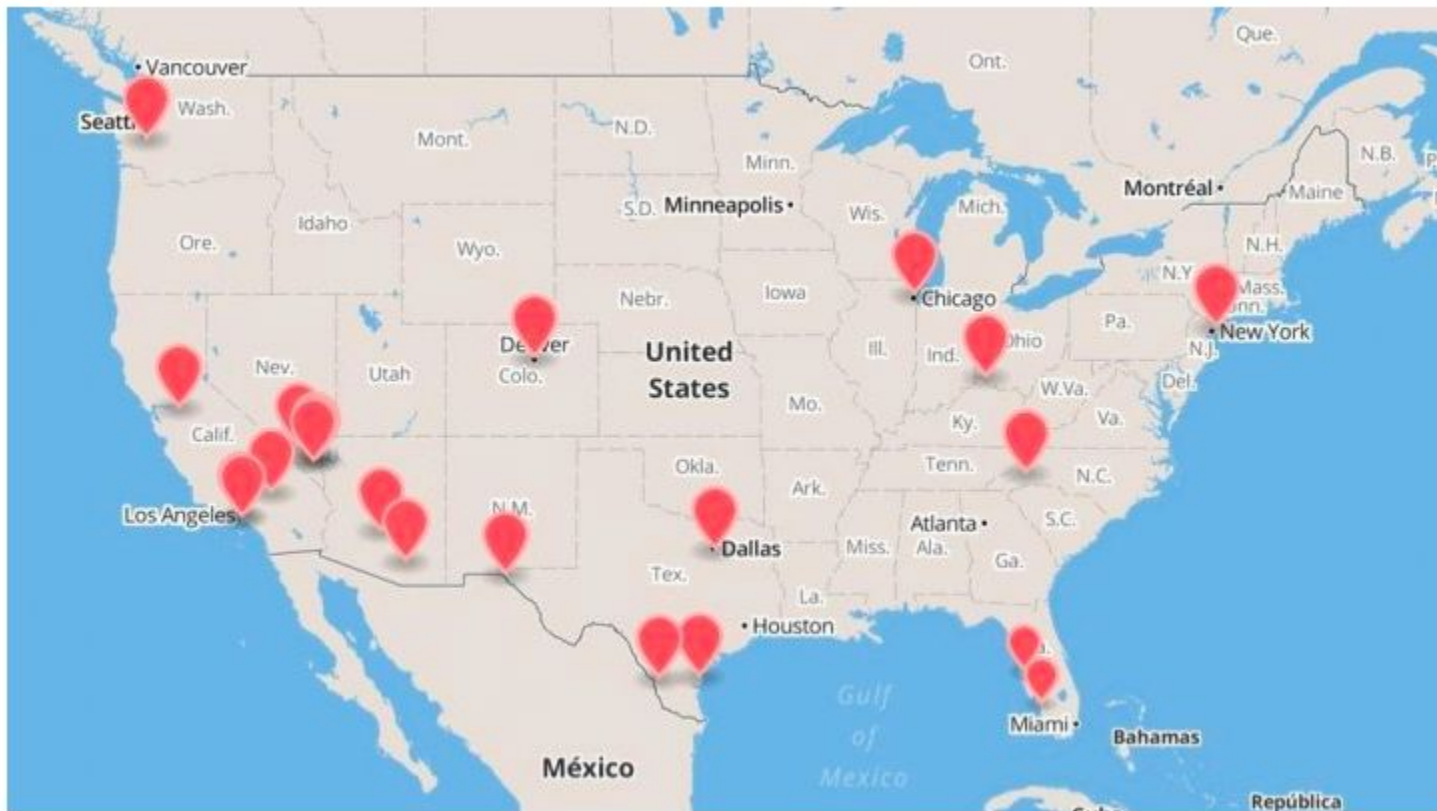


# Fake Base Station



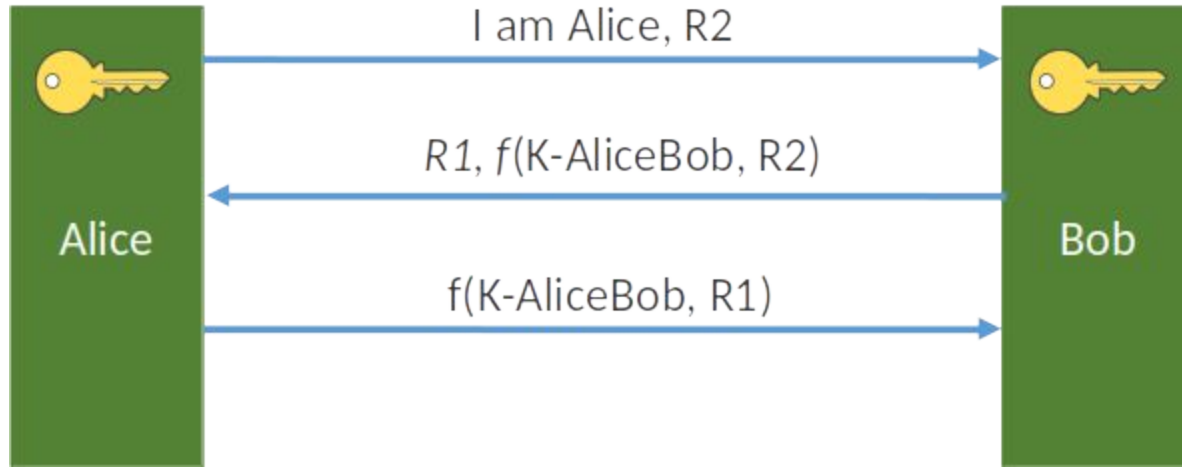
\$1500 from underground marketplace in 2013  
Send out fake/spoofed SMS with phishing links

# Fake Base Station



ESD America's map of all 19 fake mobile base station "interceptors" discovered in August 2014 (ESD America)

# Mutual Authentication with Shared Secret



# Reflection Attack



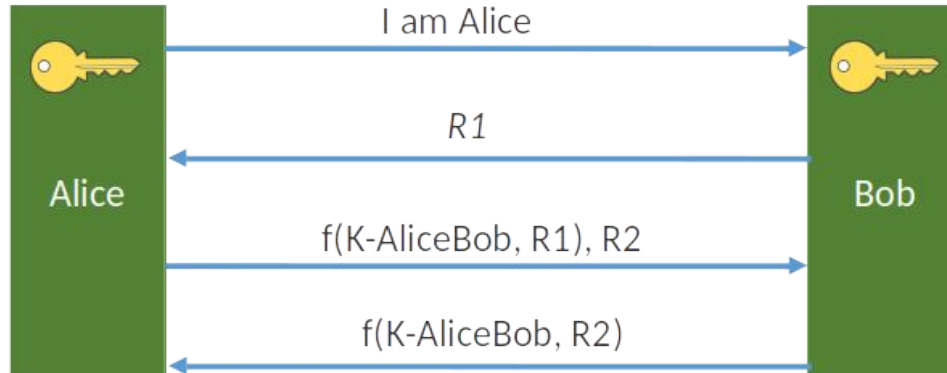
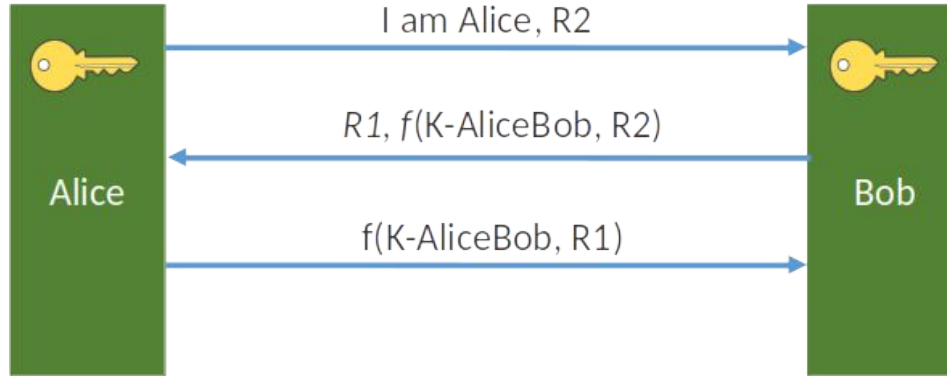
# Reflection Attack

Lesson learned:

- Don't have Alice and Bob do exactly the same thing
  - Different keys
  - Different Challenges
- The initiator should be the first to prove its identity
- Assumption: initiator is more likely to be the bad guy



# Mutual Authentication with Shared Secret



# Using Formal Methods to Verify a Protocol

## ProVerif: Cryptographic protocol verifier in the formal model

### Project participants:

[Bruno Blanchet](#), [Vincent Cheval](#)

### Former participants:

[Xavier Allamigeon](#), [Ben Smyth](#), Marc Sylvestre

ProVerif is an automatic cryptographic protocol verifier, in the formal model (so called Dolev-Yao model). This protocol verifier is based on:

- It can handle many different cryptographic primitives, including shared- and public-key cryptography (encryption and signatures), hash
- It can handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space. This result has been property is actually satisfied. The considered resolution algorithm terminates on a large class of protocols (the so-called "tagged" protocols).

ProVerif can prove the following properties:

- secrecy (the adversary cannot obtain the secret)
- authentication and more generally correspondence properties
- strong secrecy (the adversary does not see the difference when the value of the secret changes)
- [equivalences](#) between processes that differ only by terms

A survey of ProVerif with references to other papers is available at

[Bruno Blanchet](#). Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. Foundations and Trends in Privacy and Security

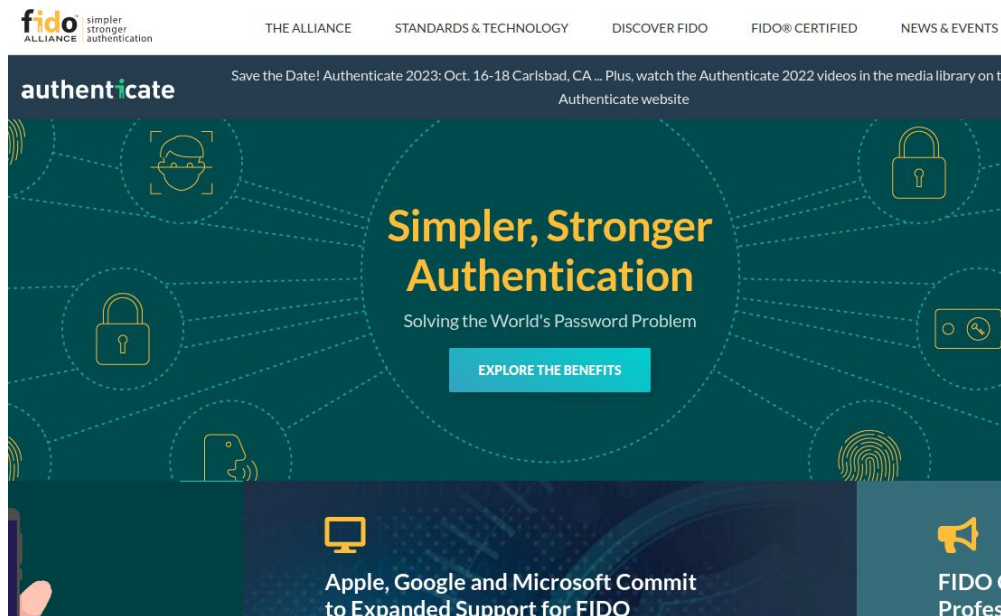
### Downloads

- To install ProVerif, you need to download:
  - Either:
    - the source package [ProVerif version 2.04 source](#) (gzipped tar file) under the [GNU General Public License](#)
    - or the binary package [ProVerif version 2.04, for Windows](#), under the [GNU General Public License](#)
  - and the documentation package [ProVerif version 2.04, documentation](#).
- [README](#) (please read this, it contains important information)
- [User manual](#) (also included in the documentation package)
- For Opam users, ProVerif can also be installed via Opam (opam install proverif).

## Proverif

- The facts in ProVerif describe what the attackers knows.
- The rules in ProVerif describe how the attacker can learn new facts...
  - ... including learning new facts by using the protocol.
- The tool then tries to apply all the rules to learn a secret.

# Fast Identity Online (FIDO)



- The FIDO protocol suite aims at allowing users to log in to remote services with a **local and trusted authenticator**
- With FIDO, relying services do not need to store **user-chosen secrets** or their **hashes**, which eliminates a major attack surface for e-business

# Fast Identity Online (FIDO)









- No secrets on the Server side
- Biometric Data (if used) never leaves device

# Fast Identity Online (FIDO)

Home / Certification Overview / FIDO® Certified

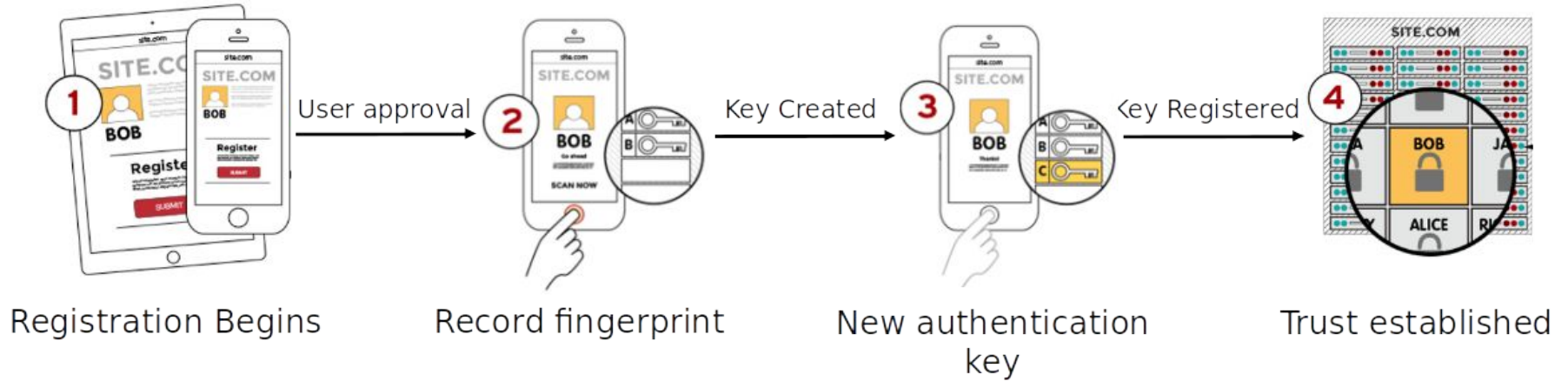
## FIDO® Certified

 <b>Company Name</b> AhnLab <b>Implementation Name</b> AhnLab FIDO® Authenticator 1.0 <b>Specification</b> UAF <b>Version</b> 1.0 <b>Type</b> Authenticator <b>Authenticator Level</b> Functional Only	 <b>Company Name</b> AiDEEP Co., Ltd. <b>Implementation Name</b> Touch xID FIDO® Android Authenticator <b>Specification</b> UAF <b>Version</b> 1.0 <b>Type</b> Authenticator <b>Authenticator Level</b> Functional Only <b>Company URL</b> <a href="http://www.aideep.ai">http://www.aideep.ai</a>
 <b>Company Name</b> AiDEEP Co., Ltd. <b>Implementation Name</b> Touch xID FIDO® Android S/W Authenticator <b>Specification</b> UAF <b>Version</b> 1.0 <b>Type</b> Authenticator <b>Authenticator Level</b> Functional Only <b>Company URL</b> <a href="http://www.aideep.ai">http://www.aideep.ai</a>	 <b>Company Name</b> AT Solutions <b>Implementation Name</b> FIDO® @SmartOneTzPIN <b>Specification</b> UAF <b>Version</b> 1.0 <b>Type</b> Authenticator <b>Authenticator Level</b> Functional Only
 <b>Company Name</b> AT Solutions <b>Implementation Name</b> FIDO® @SmartOneTzPIN <b>Specification</b> UAF <b>Version</b> 1.0 <b>Type</b> Authenticator	 <b>Company Name</b> Aware, Inc. <b>Implementation Name</b> Aware FIDO® Face Authenticator (Android) <b>Specification</b> UAF <b>Version</b> 1.0 <b>Type</b> Authenticator

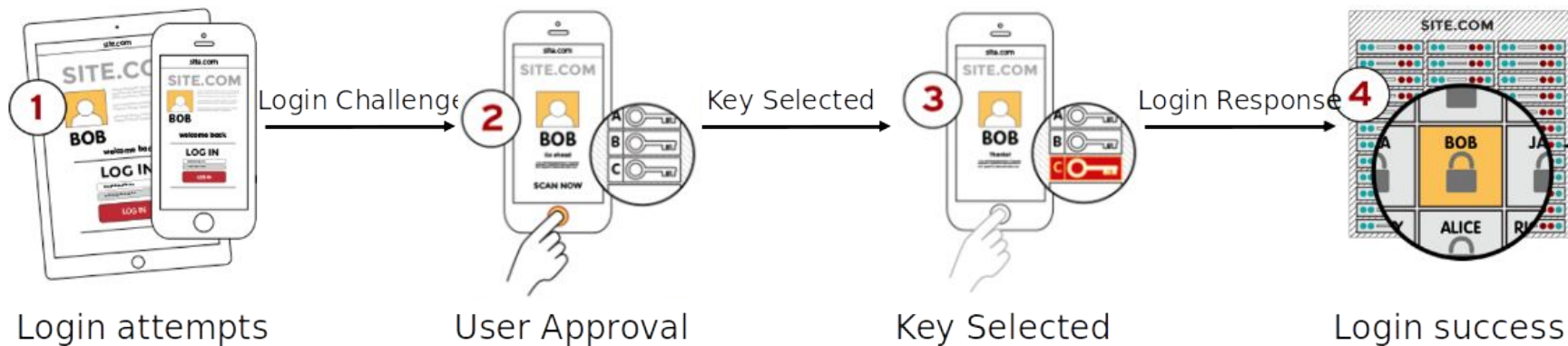
## LEADING THE EFFORT



# Fast Identity Online (FIDO)



# Fast Identity Online (FIDO)

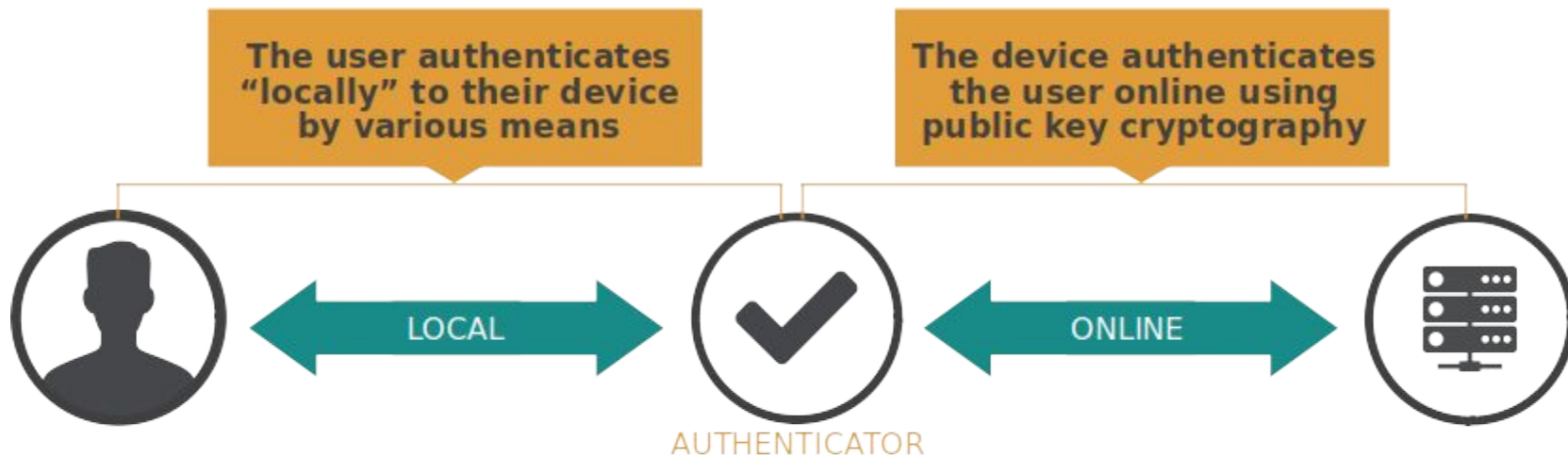


# Fast Identity Online (FIDO)

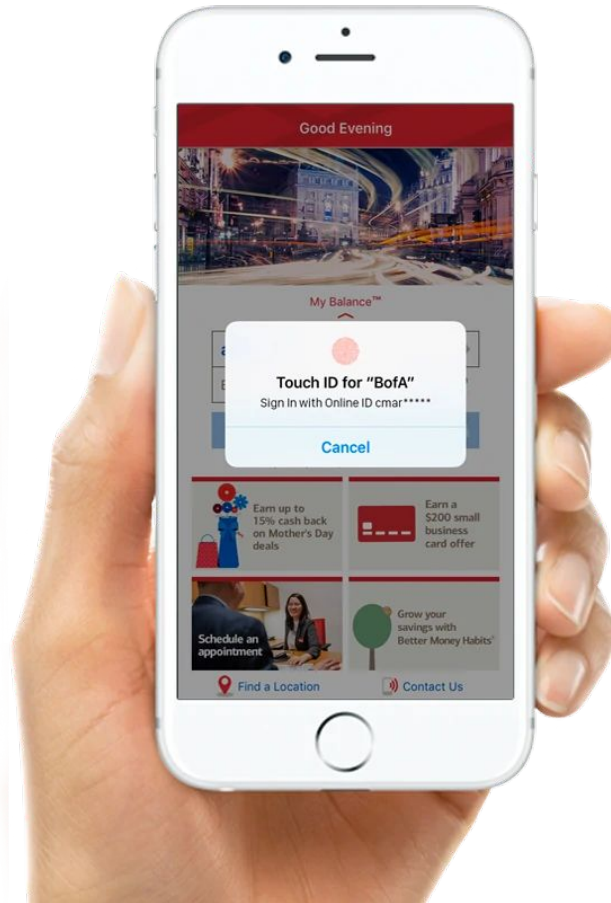
- FIDO authenticator Registration:
  - A user wishes to log in to remote services using a device that has a certified UAF authenticator, e.g., fingerprint sensor. The authenticator has a trusted attestation key (either RSA or ECDSA).
  - The user logs in to a relying party, such as a banking website, using her original credentials, e.g., text-based password. The authenticator records her fingerprint, generates an authentication key for this website, signs the public part of the new key with the attestation key, and sends it to the website.
  - The website links the user's online profile with the authentication key if it is valid. As a result, the trust between the relying party and the authenticator is established and the procedure of authenticator registration is completed.
- In subsequent login attempts (the authentication procedure), the user only needs to prove her identity to the local authenticator, upon the success of which the website and the authenticator will run a challenge-response protocol with the authentication key



# Fast Identity Online (FIDO)

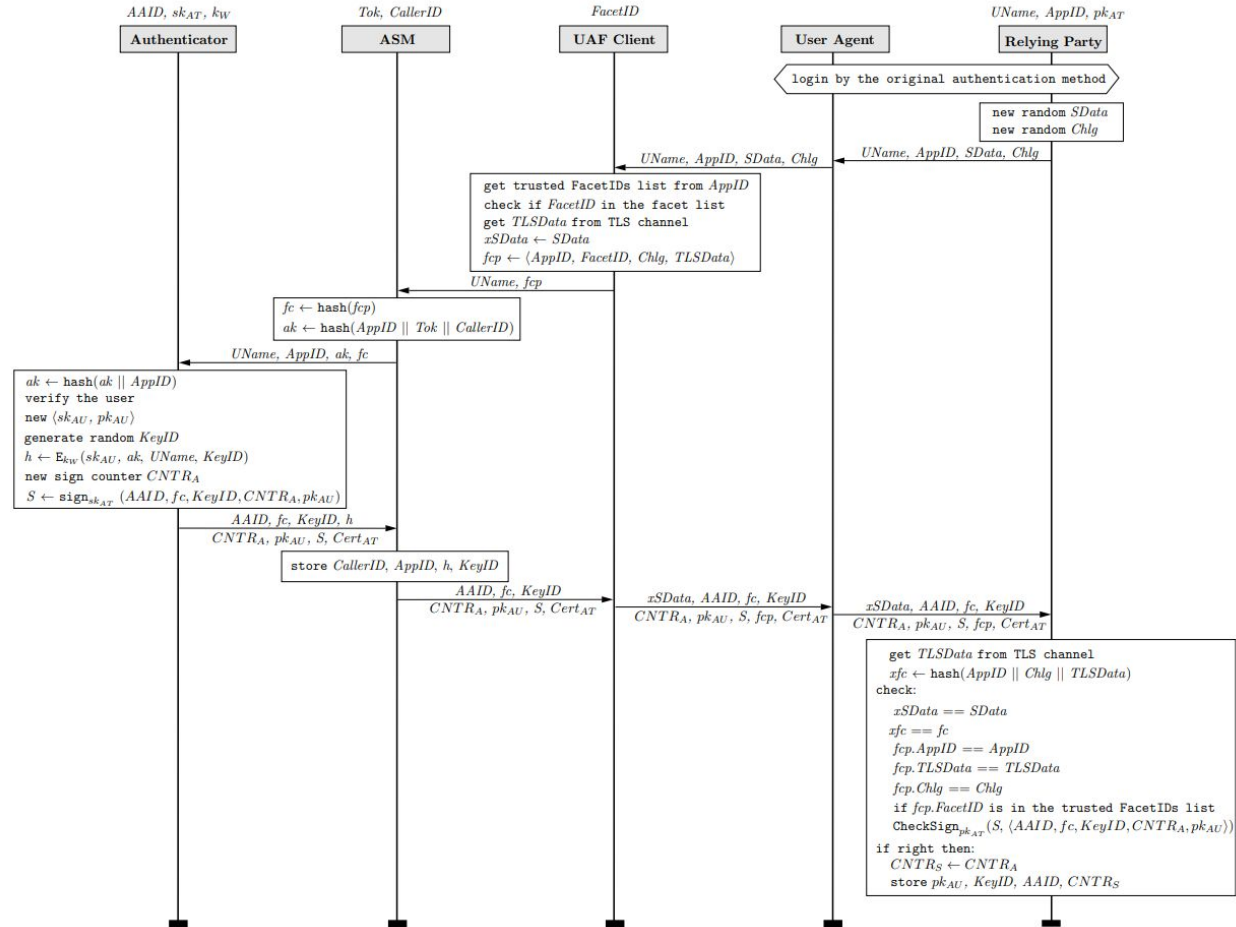


# Fast Identity Online (FIDO)



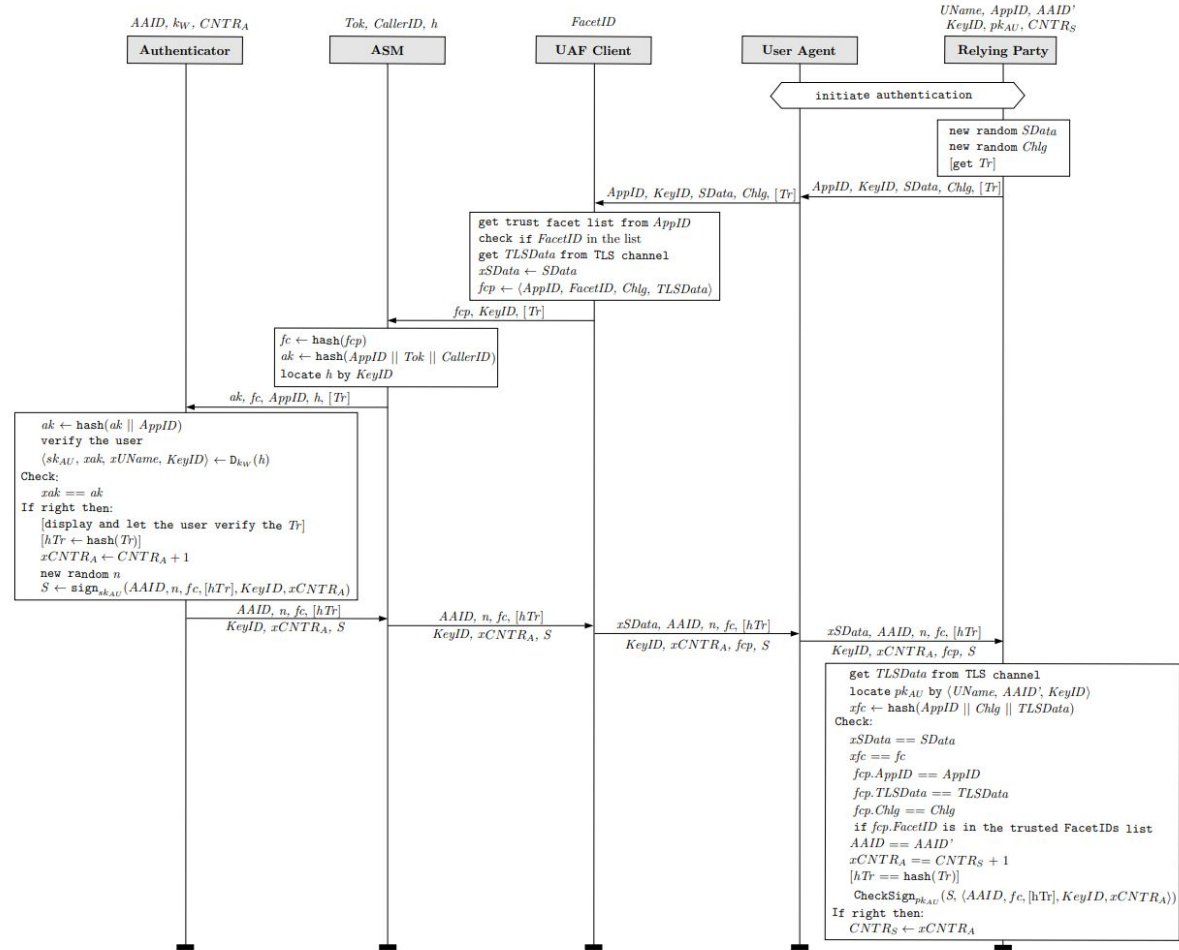
# Fast Identity Online (FIDO)

## Registration

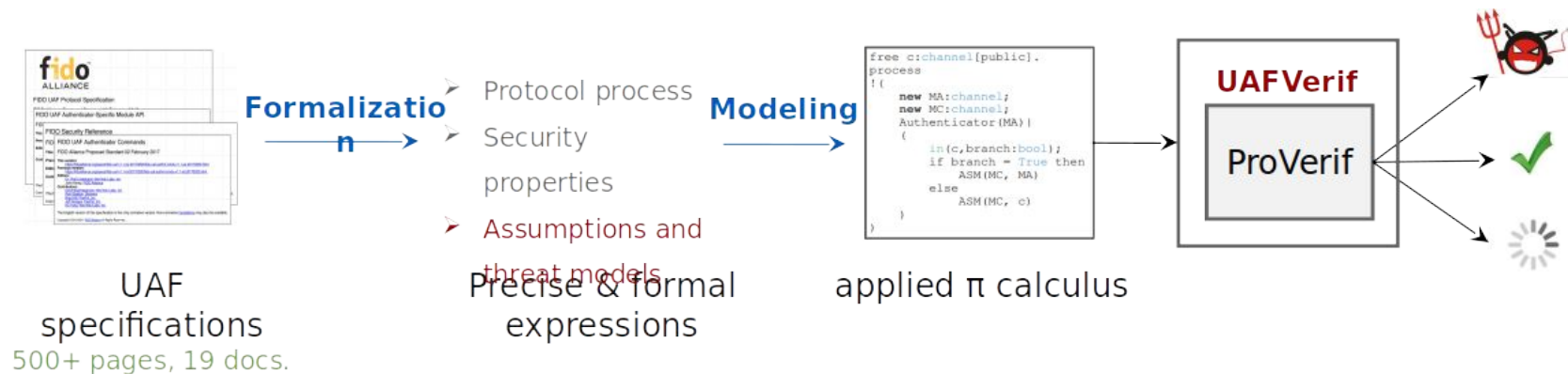


# Fast Identity Online (FIDO)

## Authentication



# Modeling FIDO using ProVerif



- **Formalization**
  - Complex description of the protocol
  - Ambiguous and implicit security properties
  - Unclear security assumptions and threat model
- **Modeling**
  - Encode expression with applied  $\pi$  calculus
  - Different threat models
- **UAFVerif**
  - Automatically change the threat model
  - Identify minimal assumptions

# Modeling FIDO using ProVerif

- Protocol process ~900LoC
  - Unbounded number of entities + unbounded sessions
  - Different threat models
- Security assumptions and security properties
  - Finding a bug of ProVerif 1.98 when analyzing authentication goals in our threat model (fixed in ProVerif 2.01)
  - Modeling unlinkability with observational equivalence
- UAFVerif ~500LoC
  - Automatically generate and analyze 400,000+ scenarios under different assumptions.
  - Automatically identify minimal assumptions
  - More than 80 hours analyzing

# Modeling FIDO using ProVerif

## Verification results

Auth.	Type	1B		2B	1R		2R
		login	step-up	step-up	login	step-up	step-up
C.	$sk_{AU}$	$\neg k_W \checkmark \neg A[M]$		$\neg k_W \checkmark \neg A[M]$	$\checkmark$		$\neg k_W$
	$ak$	$\neg tok \wedge \neg A[M]$		$\neg tok \wedge \neg A[M]$	$\times$		$\times$
	$CNTR$	$(\neg k_W \wedge \neg A[M]) \checkmark \neg M[A]$		$(\neg k_W \wedge \neg A[M]) \checkmark \neg M[A]$	$\neg C[M] \wedge \neg M[A]$		$\neg C[M] \wedge \neg M[A]$
	$Tr$	$-$	$\neg C[U] \wedge \neg M[C] \wedge \neg A[M]$	$\neg C[U] \wedge \neg M[C] \wedge \neg A[M]$	$-$	$\neg C[U] \wedge \neg M[C] \wedge \neg A[M]$	$\neg C[U] \wedge \neg M[C] \wedge \neg A[M]$
A.	Basic	$\neg A[M] \checkmark \neg M[A]$	$\checkmark$	$\checkmark$	$\neg C[M] \wedge \neg M[A]$	$\checkmark$	$\checkmark$
	Non-R	$-$	$\checkmark$	$\checkmark$	$-$	$\checkmark$	$\checkmark$

TABLE IV. MINIMAL ASSUMPTIONS REQUIRED FOR THE UAF AUTHENTICATION PROCESS TO ACHIEVE CONFIDENTIALITY PROPERTIES AND AUTHENTICATION PROPERTIES.

# Findings

- KHAccesstoken mechanism is futile
  - Attackers can easily compute the token and impersonate ASM
  - Attackers can intercept token and impersonate ASM
- Registration process is more vulnerable than authentication
  - Over 100,000 authenticators share the same attestation key and ID
  - No trust between ASM and authenticator before registration
- UAF meets unlinkability property
  - Any two relying parties cannot link the conversation to one user
- UAF prevents phishing attack
  - From malicious Relying party
  - From malicious User agent