Rethinking Authentication

Steven M. Bellovin
https://www.cs.columbia.edu/~smb

Why?

- I don't think we understand the real security issues with authentication
- Our defenses are ad hoc
- I regard this as a step towards a broader perspective, but probably not the final answer

Why Authenticate?

- Many privileges—i.e., access to assorted resources—are based on identity
- Mere assertion of identity is, of course, inadequate
- So—how should we authenticate?

The Usual Trilogy

- Something you know
- Something you have
- Something you are

The Usual Trilogy?

- Something you know
- Something you have
- Something you are

Generally the wrong way to look at it…

*These are not **fundamental** properties!*

Assertions

- Authentication is a security issue
- Authentication is a systems issue
- People are part of the authentication system

A correct solution must incorporate all of these aspects.

What Can Be Compromised?

- A user's machine?
- A user?
- An authentication server?
- A hardware security module?

What Can Be Compromised?

- A user's machine?
- A user?
- An authentication server?
- A hardware security module?
- Experience suggests that *all* of these can be compromised—lateral movement is apparently relatively easy

(Some) Authentication Types

- Passwords
- Challenge/response
  - SMS-based
- Time-based
- Cryptographic
- (Others elided in the interests of time—but they have similar properties)

Password Advice

- Pick a strong password
  - At least 17 characters, including five letters, three special characters, two emojis, two characters from a non-Latin alphabet, one from a non-human alphabet, and one character from a novel of existential angst
- Never reuse it
- Never write it down

Ancient Advice!

- This advice dates to 1979!
- No local storage
- No local computing capability
- A power user might have three logins and passwords

Not much of that is true today...

(Photo courtesy Perry Metzger)

Cryptographic Authentication: Symmetric Algorithm

- *f(k, x)* is the encryption of *x* with some per-user key *k*
  - May be done by the user with an external device
- Server does the same calculation
- User can return only a few bits of *f(k, x)*
- Note well: the server must know *k*, which exposes *k* to theft if the server is hacked

Cryptographic Authentication: Asymmetric Algorithm

- *f(k, x)* is the signature of *x* with some per-user private key *k* (or, more likely, the signature of *H(x)* for some hash function)
- Server verifies the signature
- The server does not know *k*
- User must return all of *f(k, x)*—that's too much to type, which means authentication is device-to-device, not user-to-device

Time-Based Authentication

(Photo by Alexander Klink, via Wikimedia Commons)

- Device contains a clock *t* and a secret key *k*
- The screen displays *F(k, t)*, truncated to six digits
- The server contains a file of per-user *k*

Attacks

Scenarios

- Let's look at some failure scenarios and see how various mechanisms fare
- No method is perfect
- And: it turns out there are remarkably similar failure modes between the different authentication methods

Guessing

- Obviously, passwords are vulnerable, which is why sites say "pick strong passwords"
  - Strong passwords don't help against phishing attacks
- Online? Rate-limit instead
- Offline? That follows a server compromise.
- This attack can only occur if the server has been hacked

Password guessing is only a major problem after a server compromise!

Phishing

- Passwords are obviously vulnerable
  - *Passwords are replayable*
- Most other methods are also vulnerable to active attacks—capture the credential and (ab)use it in real-time
  - For SMS-based challenge/response, use forged SS7 messages to redirect the challenge
- The attacker doesn't obtain the actual, reusable credential—but in some situations, one access is enough
- Defense: user's device *must* verify remote site's identity—but that protects all methods

Challenge/Response and Cryptographic Authentication

- Server sends *x*; user returns *f(x)*
- If there's a separate communication channel, *f(x)* can simply be *x*
- If there isn't, the user needs a key *k* and local computing power to calculate *f(k, x)*

Attacking Challenge/Response

- Subvert the communications channel
- Trick the system or user into sending *f(x)* to the wrong place
- Steal *k*
  - Can be stolen from the user or the server

Note well: these failure modes are very similar to those for passwords—and k isn't hashed!

Subverting the Channel

- The easiest challenge/response mechanism is SMS messages
- But—SMS routing is controlled by the Signaling System 7 network—and it's easy to become an SS7 speaker
- No longer just for governments!

Attacking Time-Based Authentication

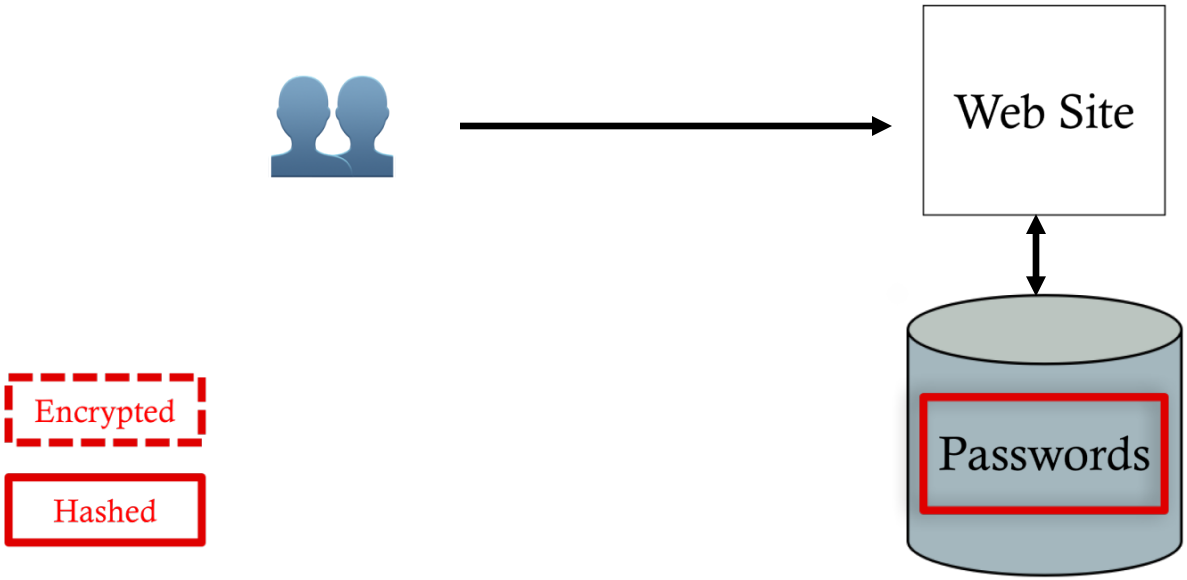
- Steal the key file from the server
- Spoof the server's idea of the time of day
- Where does the key file come from? Keys are manufactured into the devices; evidence suggests that they're either generated algorithmically or are retained by the vendor.
- Is the key file protected by an HSM? What is the API to the HSM? Is it secure? (Hint: most aren't.)

Server Compromise

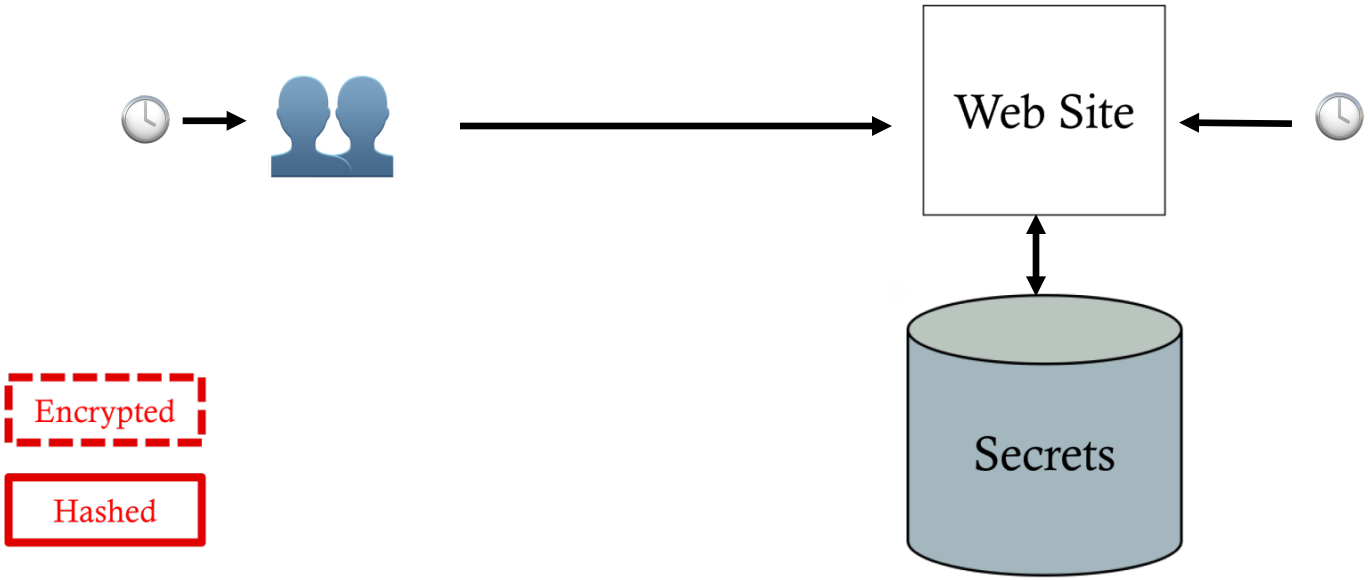
- Server compromise threatens most forms of authentication
- Challenge/response—if cryptographic, the attacker gets everyone's keys, and not just hashed passwords
- The same is true for time-based authentication and some cryptographic authentication
- Cryptographic authentication might be safe, but *only* if asymmetric crypto is used

Authentication System Architectures

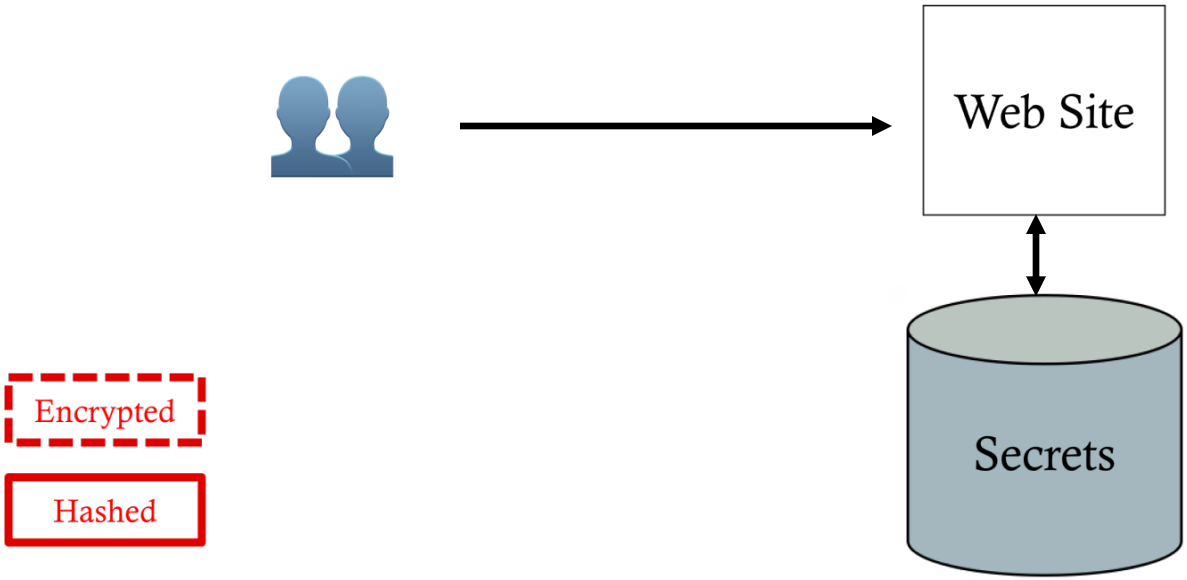
Password Authentication



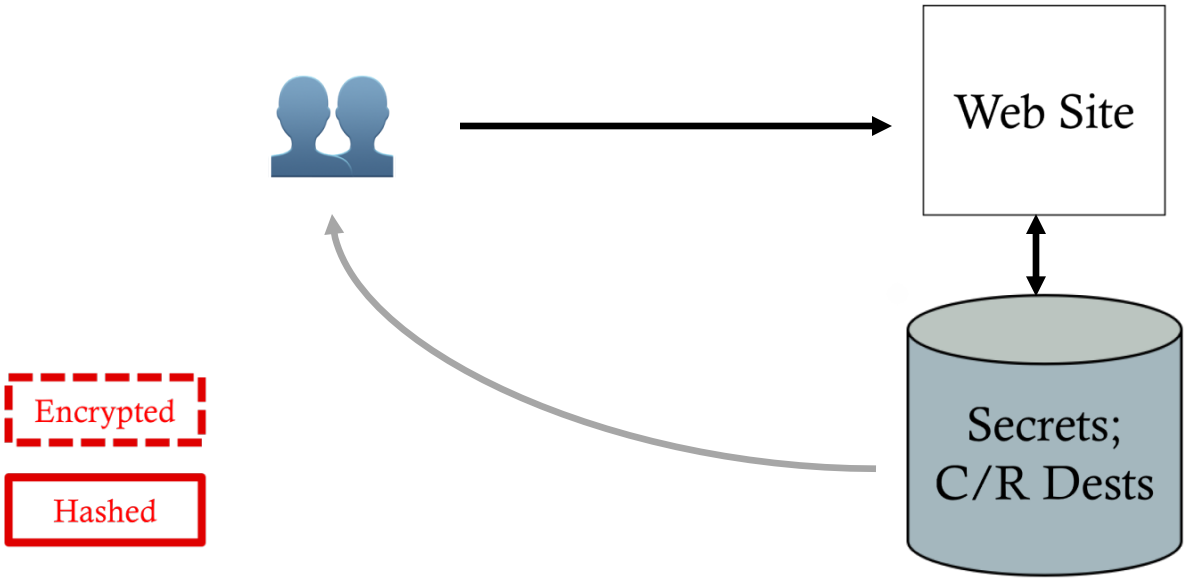
Time-Based Authentication



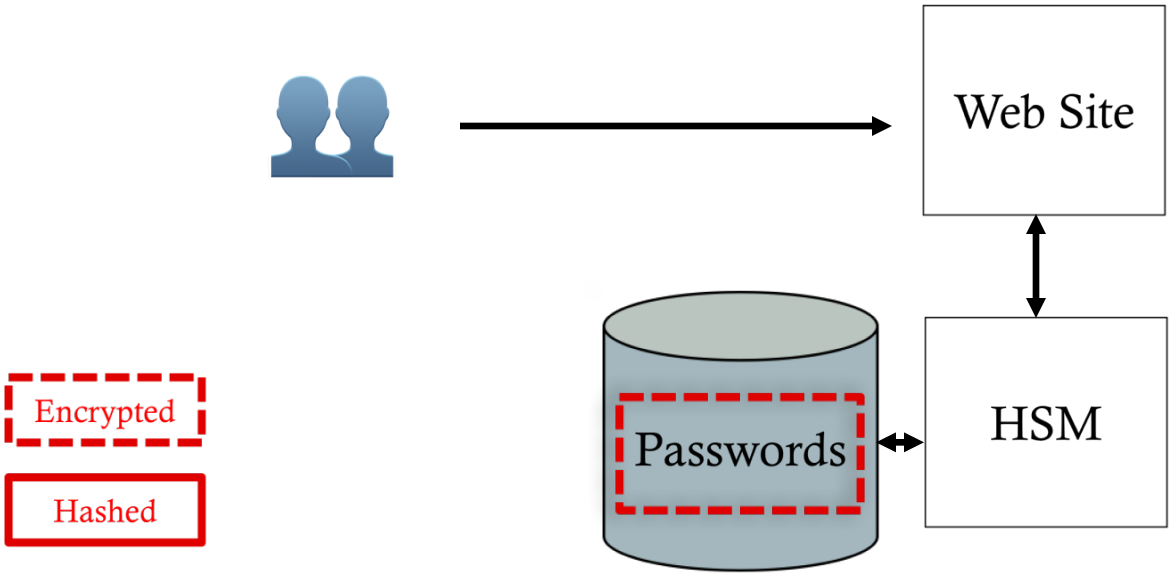
Cryptographic Authentication



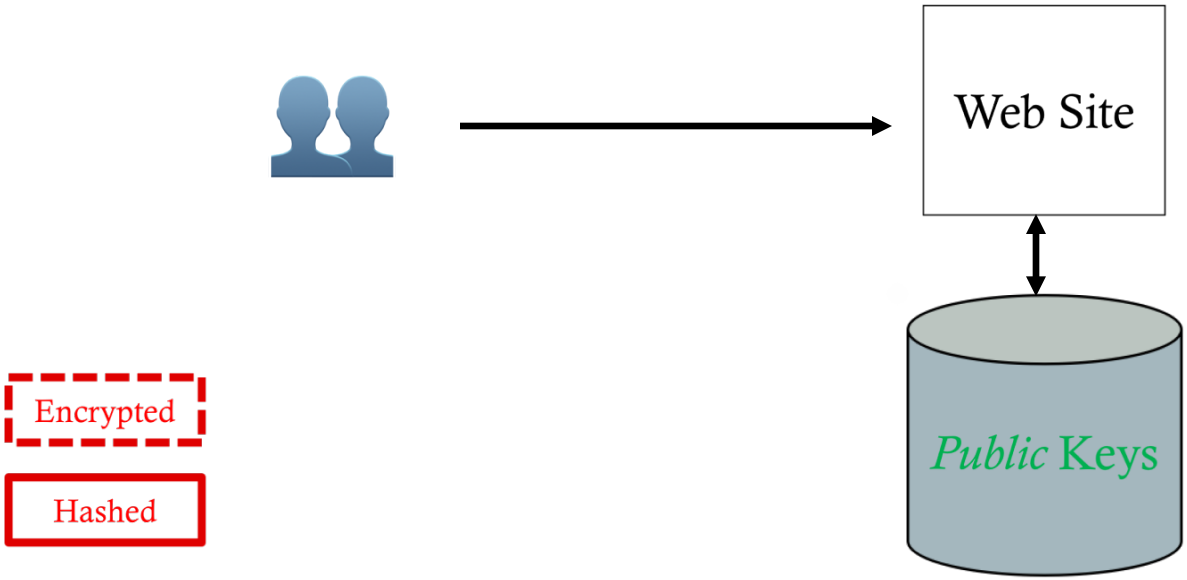
Challenge/Response Authentication



HSM-Protected Password Authentication



Asymmetric Cryptographic Authentication



Summary

Bellovin

Summary

- There are *no* perfect authentication methods
- Two-factor authentication is often stronger not because of the other mechanism's strength but because its failure modes are slightly different
  - Most important, they're (usually) not replayable over time
- Server compromise and phishing resistance are *very* difficult

Passwords versus Everything Else

- Passwords are replayable *over time* and reusable *across sites*
  - Without server authentication, other credentials are still useful to the attacker once
- Passwords are *better* than most against server compromise
  - But the defense—strong (i.e., unguessable) passwords—comes at a heavy cost in usability
- A lot of today's two-factor authentication schemes are quite weak and will fall to attackers as soon as they retool slightly

Risk Factors

	Guessing	Forgetting	Device loss	Server file compromise	Replay	External trust
Passwords	✗	✗	✓	✗	✗✗	✓
Chall/resp	✓	✓	✗	✗✗	✓	✓
SMS	✓	✓	?	✓	✓	?
Crypto	✓	✓	?	✗, ✓	✓	✓
Time-based	✓	✓	✗	✗✗	✓	✗
Biometric	✓	✓	?	✗	?	✓

✓	No particular problem; strong point of this mechanism
?	Some trouble or implementation-dependent
✗	Significant risk
✗✗	Very serious risk

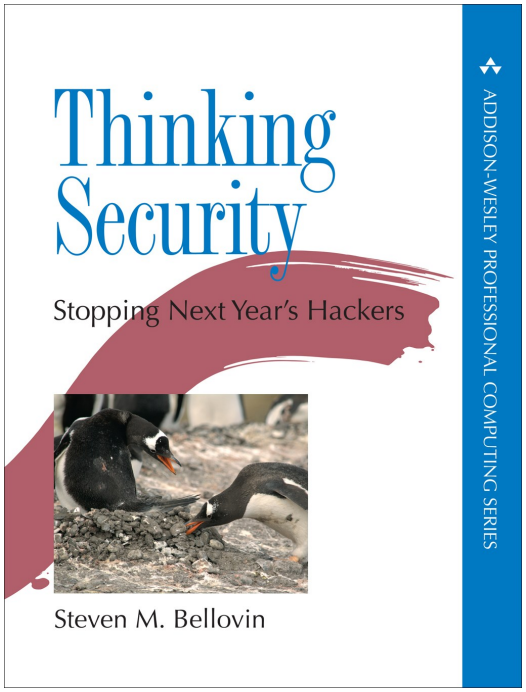
What to Do?

- Asymmetric cryptographic authentication is generally the strongest scheme
- However, it requires trusted hardware: people can't type a long signature
  - This hardware can also verify the remote site's identity
- This rules out use from other folks' devices—though that's probably a good thing

Use an iPhone?

- Computers can talk to it via secure, local channel
- It can verify the remote site's identity and use asymmetric crypto to authenticate you
- It protects secrets via its secure enclave
- Can be unlocked via pretty good biometric verification
- U2F (e.g., Yubikey) is pretty good, but lacks a display to show you what you're authorizing

Disclaimer



Much of the material in this talk comes from Chapter 7 of *Thinking Security*.

Questions?