How to use Encryption for Defense in Depth In Native and Browser Apps

November 13, 2019

Isaac Potoczny-Jones
ijones@tozny.com
@SyntaxPolice

TOZNY
https://tozny.com

When we think of driving safety…

TOZNY

…We address both the road *and* the car.

Securing the Road

- Stoplights
- Speed limits
- Gentle curves
- Lines on the road
- No passing zones

Encrypting the Infrastructure

- HTTPS / TLS
- VPNs, IPSEC
- Full Disk Encryption
- Transparent Database Encryption

TOZNY

Securing the Car

- Seatbelts
- Crumple zones
- Airbags
- Horns
- (Better driving)

Securing the Application

- Malware
- Buffer Overflows
- Side Channels
- Broken Authentication
- (Better programming)
- (Not much crypto…)

TOZNY

More Application-Level Security

Should Include Application-Level Data Encryption

Because: Defense In Depth

TOZNY

But Application-Level Means More Programming

- Many applications share the same infrastructure
  - So infrastructure-level cryptography is easier to implement
- With application-level cryptography
  - More programmers write more code
  - And we will often get it wrong
- The encryption community isn't great about making developer tools
  - This is the subject of another series of talks

As a result, we don't do it, even though we should

TOZNY

COMPUTERWORLD
FROM IDG

Software developers aren't implementing encryption correctly

Lack of specialized training for developers and crypto libraries that are too complex lead to widespread encryption failures

"Developers are adding a lot of crypto to their code, especially in sectors like health care and financial services, but they're doing it poorly"

- Veracode CTO Chris Wysopal.

3 Cryptographic and credentials management problems plague code.

Crypto issues follow closely on the heels as the second most common type of vulnerability, and credentials problems are in the top 10.

About **35%** of applications use hard-coded passwords

and **39%** use broken or risky crypto algorithms.

TOZNY

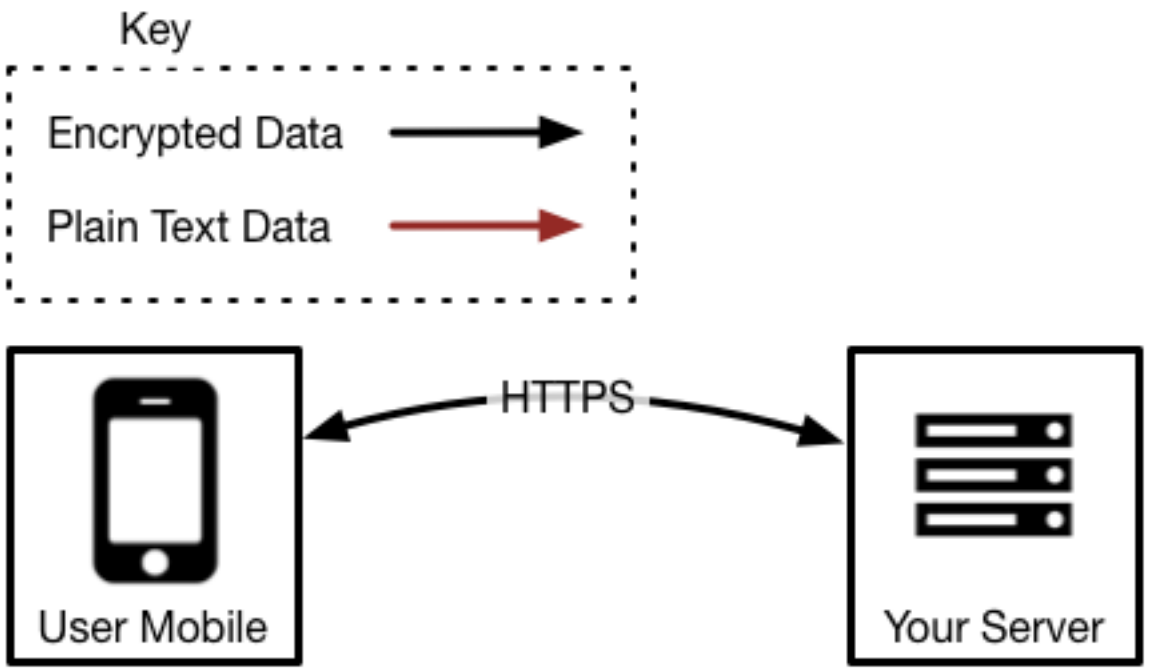
Infrastructure-Layer Encryption

In the Browser*

*(HTTPS)

TOZNY

Conceptual HTTPS

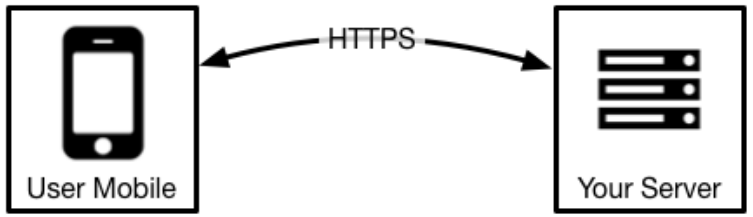


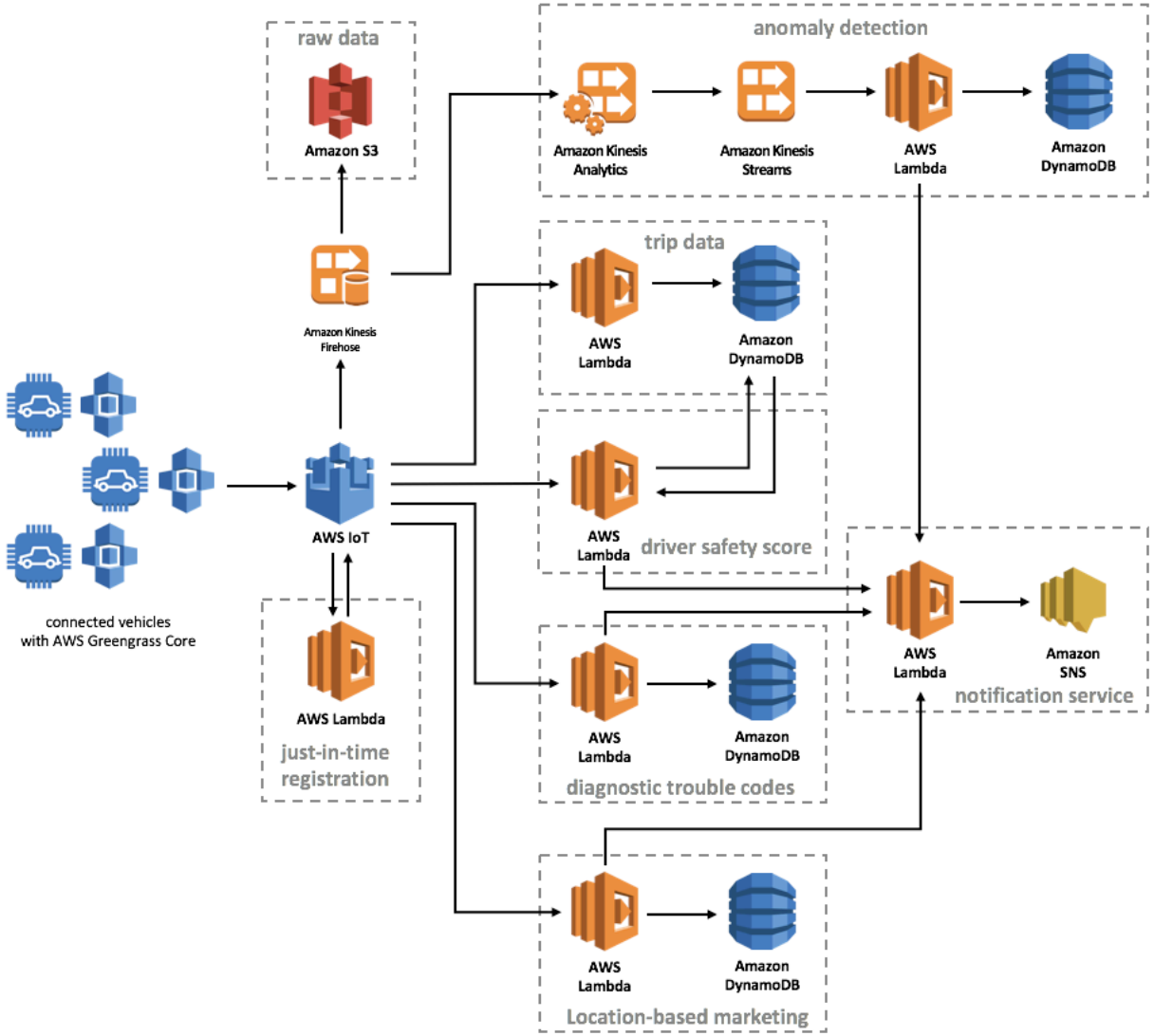
TOZNY

A more realistic system architecture

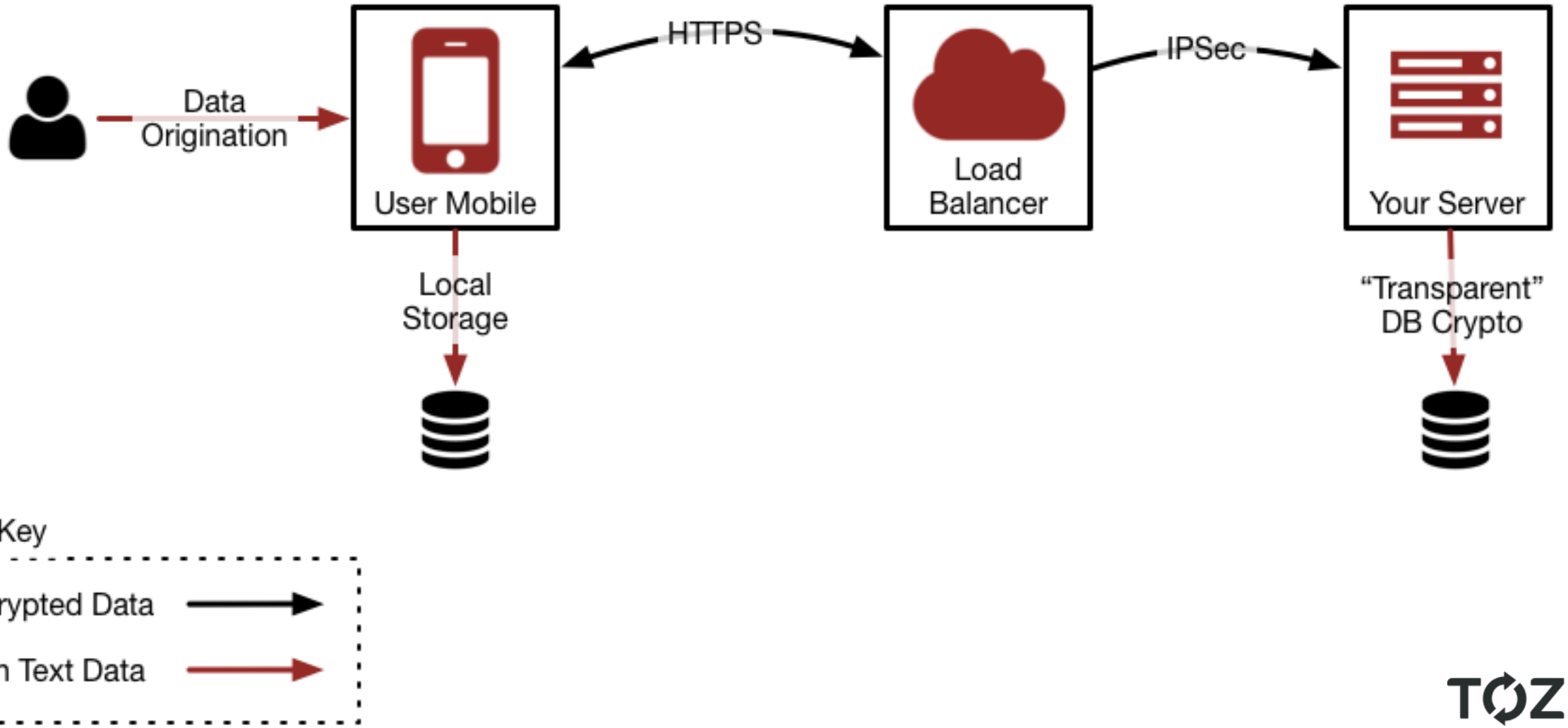
TOZNY

Oversimplifying the view of infrastructure makes infrastructure-level encryption seem easier





Actual HTTPS (Plus IPSec and DAR)



TOZNY

169 Trusted Certificate Issuers in MacOS

TOZNY

Certificate name
AAA Certificate Services
Actalis Authentication Root CA
AddTrust Class 1 CA Root
AddTrust External CA Root
Admin-Root-CA
AffirmTrust Commercial
AffirmTrust Networking
AffirmTrust Premium ECC
AffirmTrust Premium
Amazon Root CA 1
Amazon Root CA 2
Amazon Root CA 3
Amazon Root CA 4
ANF Global Root CA
Apple Root CA - G2
Apple Root CA - G3
Apple Root CA
Apple Root Certificate Authority
ApplicationCA2 Root
Atos TrustedRoot 2011
Autoridad de Certificacion Firmaprofesional CIF A62634068
Autoridad de Certificacion Raiz del Estado Venezolano
Baltimore CyberTrust Root
Belgium Root CA2
Buypass Class 2 Root CA
Buypass Class 3 Root CA
CA Disig Root R1
CA Disig Root R2
Certigna
Certinomis - AutoritÃ© Racine
Certinomis - Root CA
Certplus Root CA G1
Certplus Root CA G2
certSIGN ROOT CA
Certum CA
Certum Trusted Network CA 2
Certum Trusted Network CA
CFCA EV ROOT
Chambers of Commerce Root - 2008
Chambers of Commerce Root
Cisco Root CA 2048
Class 2 Primary CA
COMODO Certification Authority
COMODO ECC Certification Authority
COMODO RSA Certification Authority
ComSign CA
ComSign Global Root CA
ComSign Secured CA
D-TRUST Root CA 3 2013
D-TRUST Root Class 3 CA 2 2009
D-TRUST Root Class 3 CA 2 EV 2009
Deutsche Telekom Root CA 2
DigiCert Assured ID Root CA
DigiCert Assured ID Root G2
DigiCert Assured ID Root G3
DigiCert Global Root CA
DigiCert Global Root G2
DigiCert Global Root G3
DigiCert High Assurance EV Root CA
DigiCert Trusted Root G4
DST Root CA X3
DST Root CA X4
E-Tugra Certification Authority
Echoworx Root CA2
EE Certification Centre Root CA
Entrust Root Certification Authority - EC1
Entrust Root Certification Authority - G2
Entrust Root Certification Authority
Entrust.net Certification Authority (2048)
Entrust.net Certification Authority (2048)
ePKI Root Certification Authority
Federal Common Policy CA
GeoTrust Global CA
GeoTrust Primary Certification Authority - G2
GeoTrust Primary Certification Authority - G3
GeoTrust Primary Certification Authority
Global Chambersign Root - 2008
Global Chambersign Root
GlobalSign Root CA
GlobalSign
GlobalSign
GlobalSign
GlobalSign
Go Daddy Class 2 Certification Authority
Go Daddy Root Certificate Authority - G2
Government Root Certification Authority
Hellenic Academic and Research Institutions RootCA 2011
Hongkong Post Root CA 1
I.CA - Qualified Certification Authority, 09/2009
IdenTrust Commercial Root CA 1
IdenTrust Public Sector Root CA 1
ISRG Root X1
Izenpe.com
Izenpe.com
KISA RootCA 1
Microsec e-Szigno Root CA 2009
NetLock Arany (Class Gold) FÅ‘tanÃºsÃ-tvÃ¡ny
Network Solutions Certificate Authority
OISTE WISeKey Global Root GA CA
OISTE WISeKey Global Root GB CA
OpenTrust Root CA G1
OpenTrust Root CA G2
OpenTrust Root CA G3
QuoVadis Root CA 1 G3
QuoVadis Root CA 2 G3
QuoVadis Root CA 2
QuoVadis Root CA 3 G3
QuoVadis Root CA 3
QuoVadis Root Certification Authority
Secure Global CA
SecureTrust CA
Security Communication EV RootCA1
Security Communication RootCA1
Security Communication RootCA2
Sonera Class2 CA
Staat der Nederlanden EV Root CA
Staat der Nederlanden Root CA - G2
Staat der Nederlanden Root CA - G3
Starfield Class 2 Certification Authority
Starfield Root Certificate Authority - G2
Starfield Services Root Certificate Authority - G2
StartCom Certification Authority G2
StartCom Certification Authority
StartCom Certification Authority
Swisscom Root CA 1
Swisscom Root CA 2
Swisscom Root CA 2
Swisscom Root EV CA 2
SwissSign Gold CA - G2
SwissSign Gold Root CA - G3
SwissSign Platinum CA - G2
SwissSign Platinum Root CA - G3
SwissSign Silver CA - G2
SwissSign Silver Root CA - G3
Symantec Class 1 Public Primary Certification Authority - G4
Symantec Class 1 Public Primary Certification Authority - G6
Symantec Class 2 Public Primary Certification Authority - G4
Symantec Class 2 Public Primary Certification Authority - G6
Symantec Class 3 Public Primary Certification Authority - G4
Symantec Class 3 Public Primary Certification Authority - G6
SZAFIR ROOT CA
T-TeleSec GlobalRoot Class 2
T-TeleSec GlobalRoot Class 3
TeliaSonera Root CA v1
thawte Primary Root CA - G2
thawte Primary Root CA - G3
thawte Primary Root CA
TRUST2408 OCES Primary CA
Trustis FPS Root CA
TWCA Global Root CA
TWCA Root Certification Authority
UCA Global Root
UCA Root
USERTrust ECC Certification Authority
USERTrust RSA Certification Authority
UTN - DATACorp SGC
UTN-USERFirst-Client Authentication and Email
UTN-USERFirst-Hardware
UTN-USERFirst-Object
VeriSign Class 1 Public Primary Certification Authority - G3
VeriSign Class 2 Public Primary Certification Authority - G3
VeriSign Class 3 Public Primary Certification Authority - G3
VeriSign Class 3 Public Primary Certification Authority - G4
VeriSign Class 3 Public Primary Certification Authority - G5
VeriSign Universal Root Certification Authority
Visa eCommerce Root
Visa Information Delivery Root CA
VRK Gov. Root CA
XRamp Global Certification Authority

Attack Models (HTTPS is great, but)

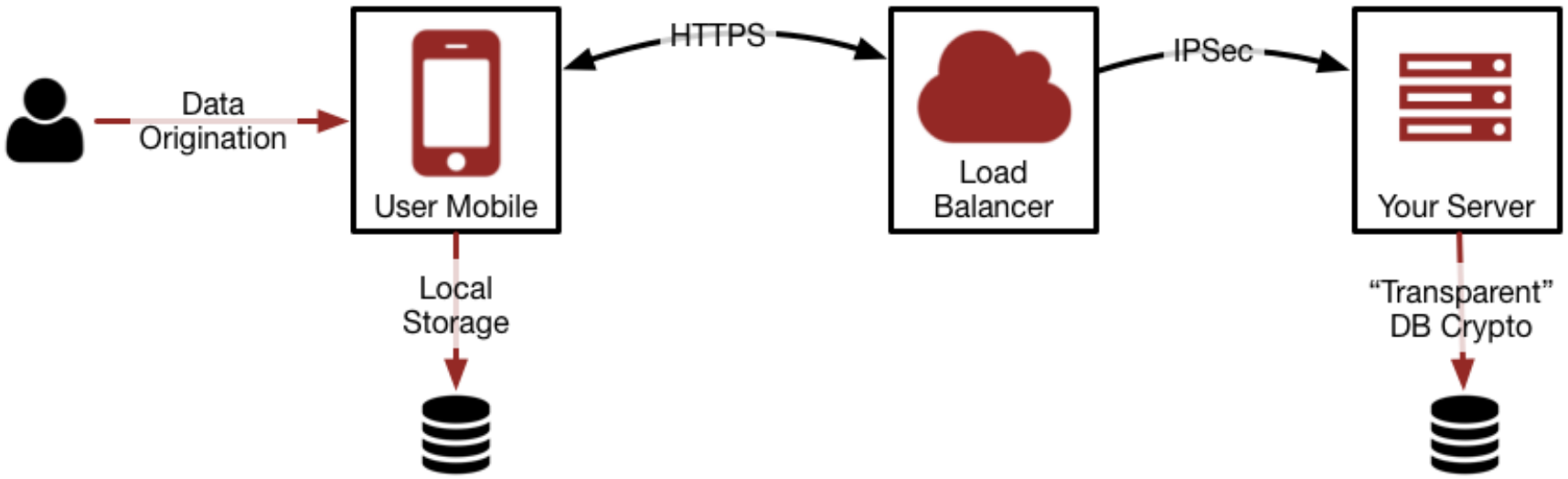
HTTPS has been attacked many times

- Subpoena the company storing the data
- Break into one of the systems NOT secured by HTTPS
- Misconfiguration of HTTPS
- Steal a key / Subvert a CA
- MITM with an enterprise-issued cert
- Trick the user by getting them to visit a different site
- JavaScript attack model
- etc...

TOZNY

Summary: Why HTTPS is not enough

- It doesn’t protect the data everywhere it goes
- It relies on a massive trust infrastructure outside user control
- It’s necessary, but not sufficient for strong privacy & security



TOZNY

Application-Layer Encryption

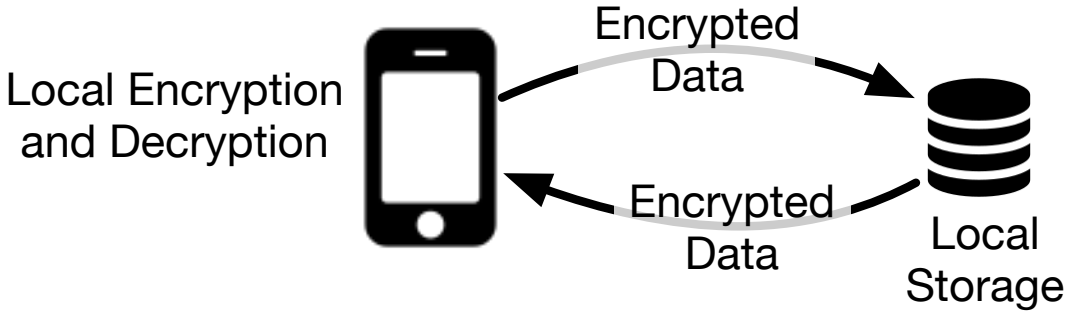
Introduction

TOZNY

One-Party / One-Device Encryption

- Encryption and decrypting is happening:
  - On the same device / in the same place / by the same person
  - E.g. local encryption of data on Android
  - Tozny publishes a popular open source library for Android for this

- This is *relatively* easy
  - Derive a key from the user's password, no need to share keys
  - Biometric on modern mobile operating systems
  - Encrypt and decrypt in the same programming language

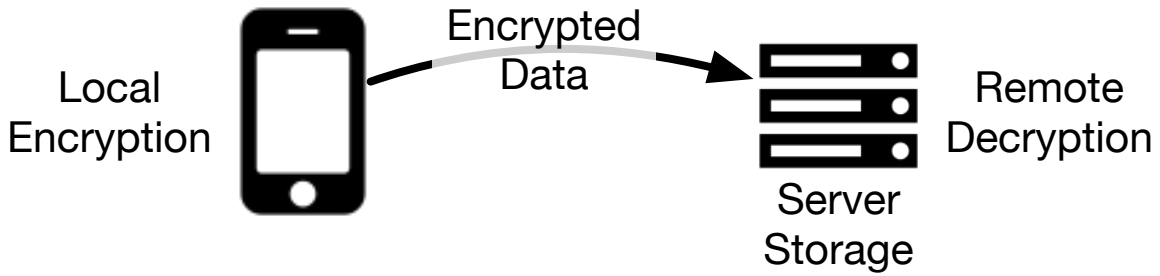


TOZNY

Communicating between users or from user to server

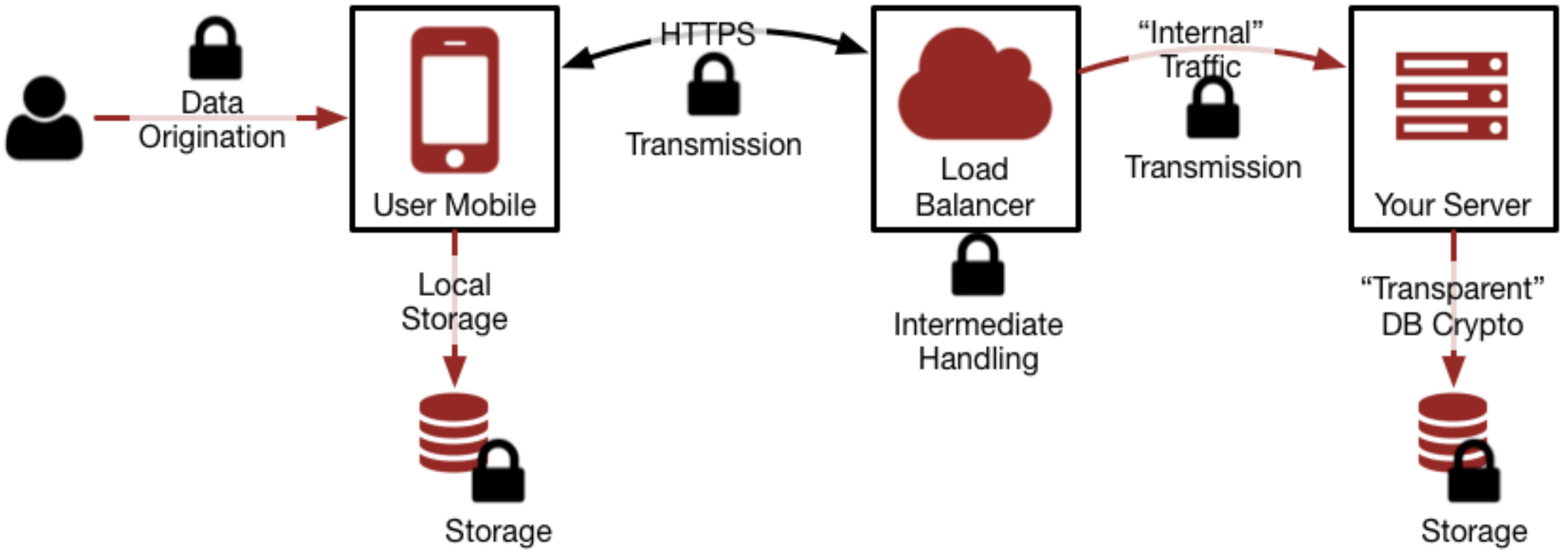
Why is this hard?

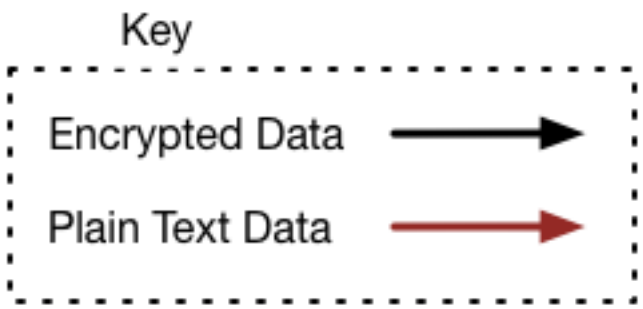
- Encrypting and decrypting in different programming languages
  - (Good luck getting all AES parameters to line up, or use libSodium)
- Various communication attacks that HTTPS takes care of for you
  - (you probably need HTTPS too)
- Key exchange / establishing trust and identity of user / device / server
  - (Users can't key)



TOZNY

Application-Layer Plus HTTPS





TOZNY

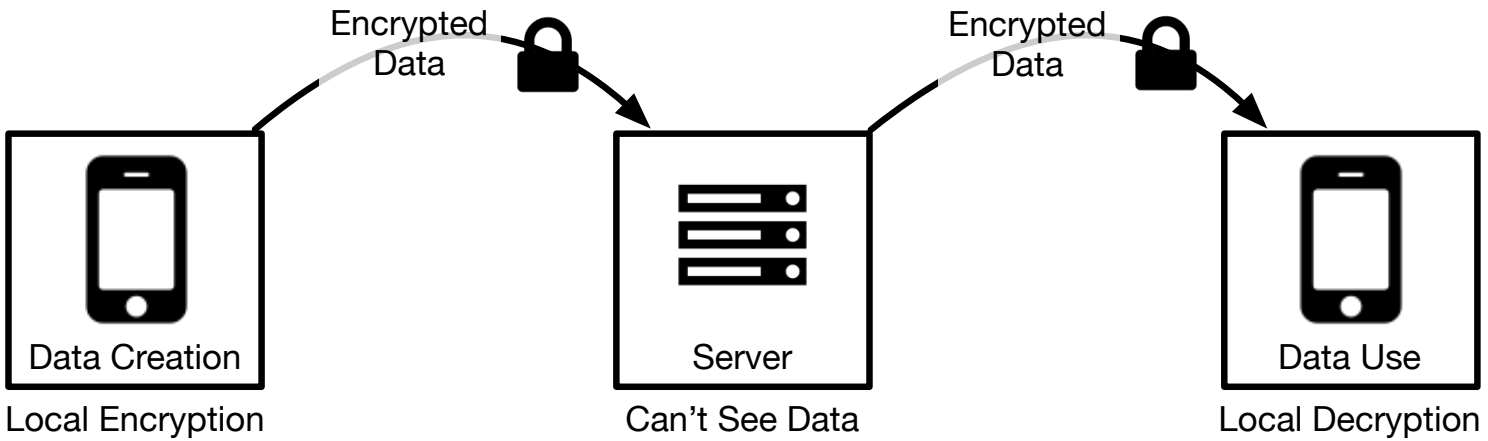
End-to-end

Encryption

TOZNY

What is End-to-end Encryption?

- Data is encrypted on the device of the person who created the data
- Data is only decrypted by the intended receiver of the data
- No intermediate parties have the keys
  - No servers, no networks, no wireless protocols, no government



TOZNY

Pretty Good Privacy / GPG

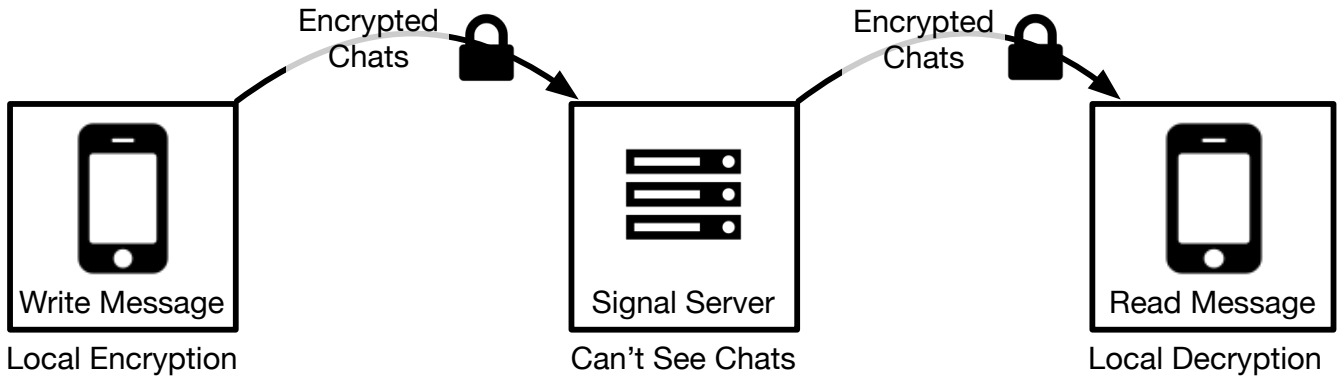
- PGP is the venerable Swiss Army Knife of encryption
- It'll encrypt anything but you have to find a way to send it
- It lets you mark public keys of other users as trusted
- We used to have parties where we would sign each-others' keys
- It's very hard to use, and even harder to use right
- Lots of stuff is built on it

TOZNY

Chat Apps

- A modern phenomenon is to have encrypted chat apps
  - Encrypt by the sender, decrypt by the receiver
  - Examples: Signal, WhatsApp, iMessage

- The goal is that people can communicate securely!
  - Everyday conversations between people who don't want to be spied on
  - People who want to share information with reporters
  - Government employees sometimes use them instead of their official email

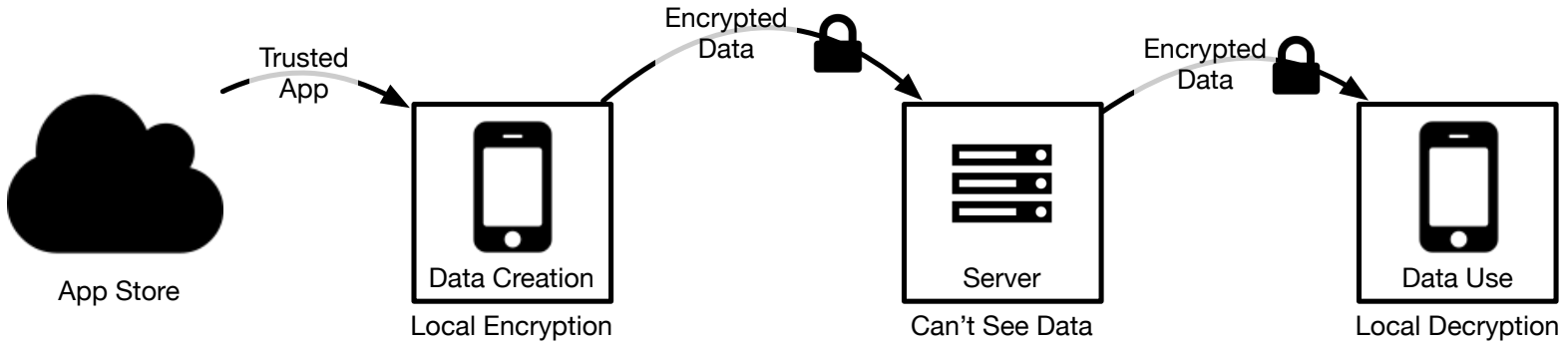


TOZNY

Applications and Code Delivery

- Encryption in mobile and desktop applications have a good workflow
  - Your phone installs an app from the app store (which has digital signatures)
  - That app has encryption / decryption code in it
  - Presumably it's vetted and the vetted app is the one that was installed
  - You use some out-of-band mechanism like a voice call to verify keys

- And unlike browser-based apps:
  - Code doesn't change on-the-fly every time you open the app
  - Your copy of the code is the same as everyone else's



TOZNY

Attacking End-to-end encryption

- It’s is so powerful that it solves* a bunch of vulnerabilities
  - Encryption for privacy, not just security
  - Subpoenas, warrants, and other government intervention against cloud
  - Misuse of data and other privacy violations
  - Undermining of HTTPS by CAs, governments, employers, etc.

- Attacks
  - Make it illegal, or target people who use e2e encryption
  - Force the developer to undermine the encryption
  - Guess the password works if the key is derived from a password
  - Best attack is to *hack the end device* (this has been happening)

*mitigates

TOZNY

Hacking the end device

"~~WhatsApp's End-to-End Encryption Is A Gimmick~~" –Bloomberg (May '19)

- Without end-to-end encryption, bad guys can use dragnet attacks
  - Get all of the data from hacking the database
  - Use that against whoever you find

- With end-to-end encryption, bad guys must use targeted attacks
  - This is a very good thing! It's what you want!
  - Exploits against end user devices get discovered and fixed

Celebrate when you see headlines saying "encrypted app got hacked"!

TOZNY

Application-Layer Encryption

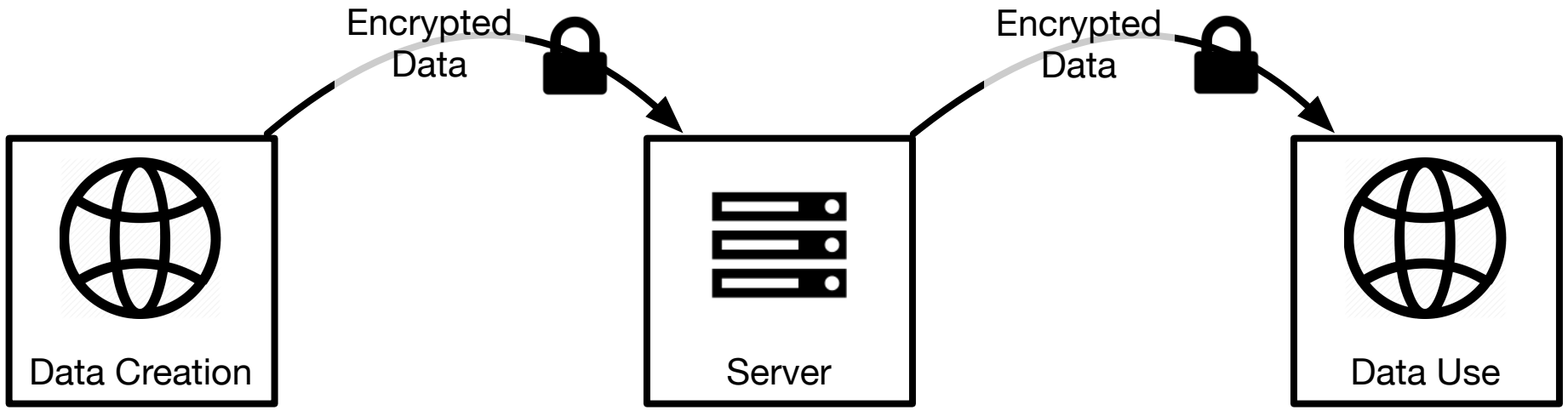
In the Browser*

*(Don’t freak Out)

TOZNY

What it is

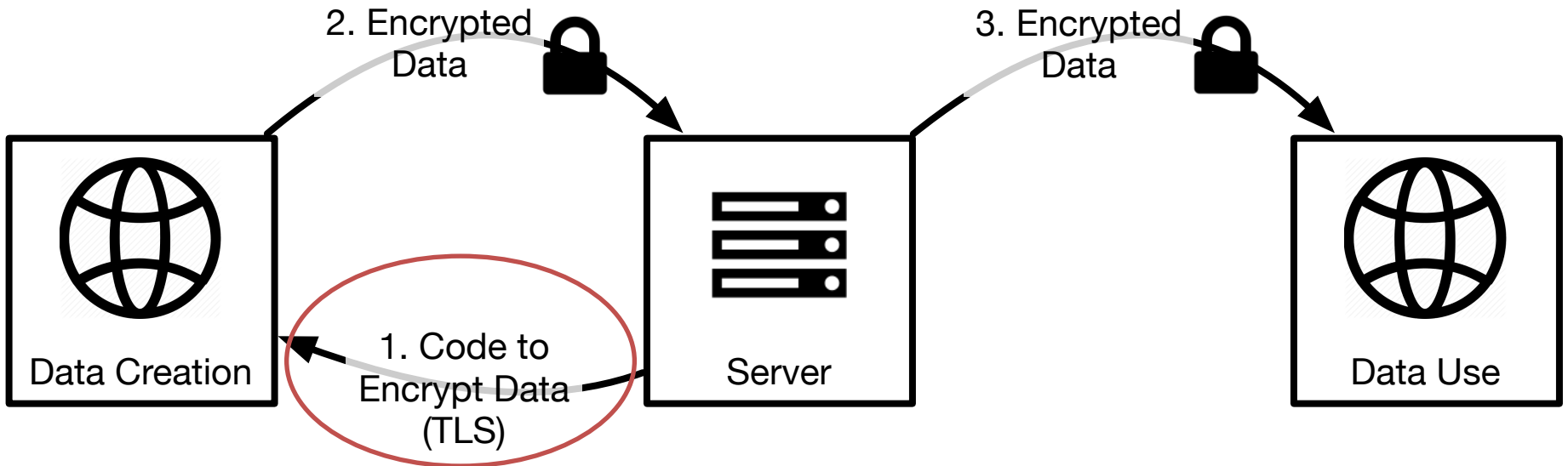
- Ship encryption code to the browser, probably in JavaScript
- Use password-derived keys or local / session storage to manage keys
- Encrypt data with JavaScript before sending it on its way
- This is how ProtonMail works



TOZNY

Attacking Browser-Layer Encryption

- Crypto code gets dynamically shipped to the browser
  - That code can exfiltrate the data
  - Can target specific users
  - Can change moment to moment
  - How can you trust it?
  - e.g. Attack the TLS connection or modify the code before it goes to the user



TOZNY

A good paper on the topic

An Analysis of the ProtonMail Cryptographic Architecture

Nadim Kobeissi
Symbolic Software
nadim@symbolic.software

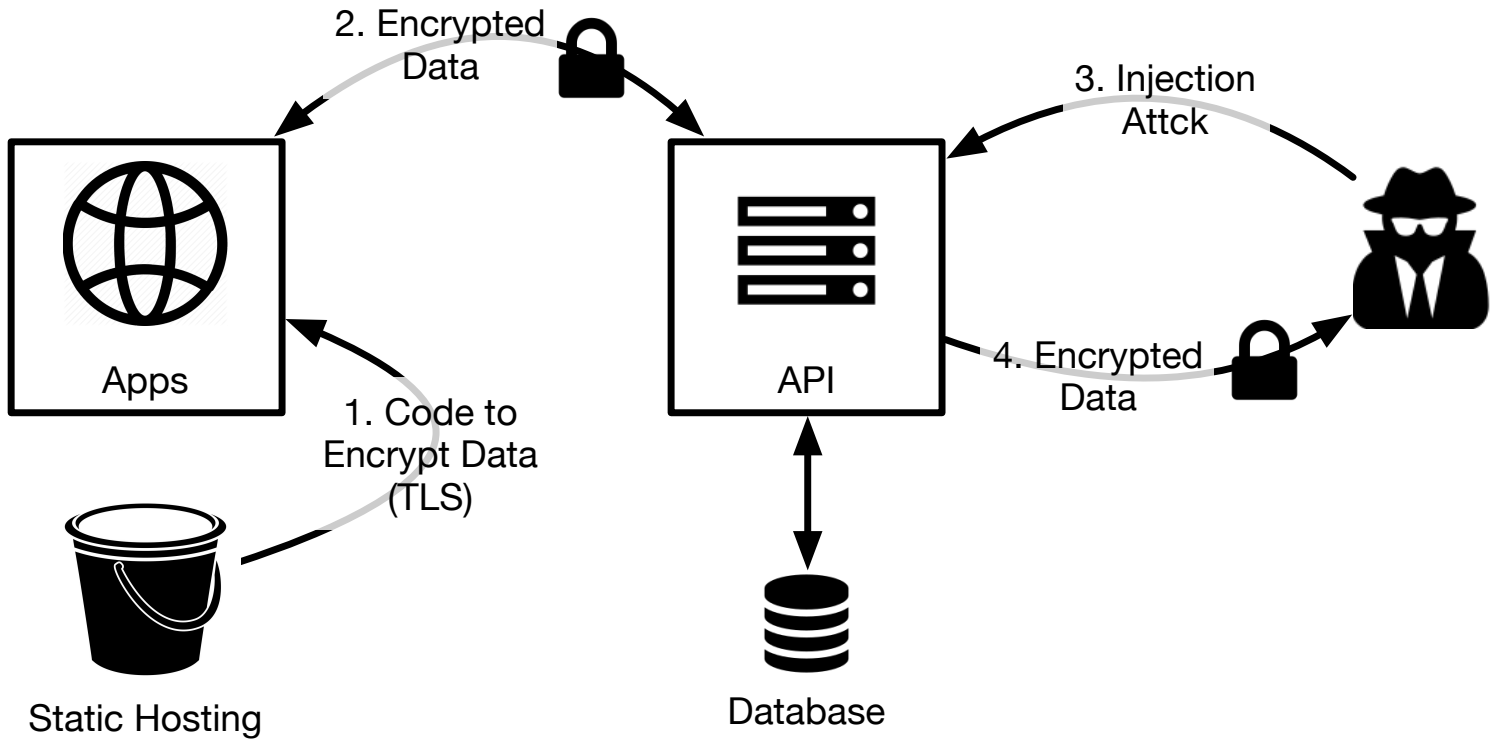
November 27, 2018

https://eprint.iacr.org/2018/1121.pdf

TOZNY

But remember that real services are more complex

- What if your static hosting bucket is secure
- But you have an SQL injection in your API code?
- Attacks aren't "all or nothing"



TOZNY

”It’s Broken Anyway” is a problematic mindset

It argues that:

- End users shouldn’t use the web for sensitive stuff
- It’s impossible to do end-to-end encryption in the browser
- It’s not really end-to-end encryption
- It relies on TLS anyway

TOZNY

Defense in Depth: Application-Layer crypto mitigates problems*

- It reduces the need to trust your entire infrastructure
  - The API server and the admins on that server don't see the plain text
  - The plain text isn't in the database and subject to e.g. SQL injections
  - Data has a way of spreading, e.g. going into backups unencrypted
  - ”Web servers” are not one thing anymore; infrastructure is complex, and attacks are not “all or nothing”
- It's harder to force a company to change code than to hand over data
- Attacks are detectable on the client, increasing risk of getting caught
- Browsers are improving in the ability to trust code
- You can add application-level access rules and enforcement logic

(*Even in the browser)

TOZNY

You should do browser crypto anyway

- Most people accept most default settings. Default security = security
  - Lots of people use browsers for sensitive stuff. Make it secure
  - Corollary: ProtonMail is a good idea; webmail is very popular
- Can browser crypto really be “end-to-end encryption”?
  - I’d say ”yes” but reasonable people can disagree
  - Be sure you understand the security claims you’re making
- We shouldn’t be pretending that e2e is perfect security anyway
  - See WhatsApp got hacked narrative

TOZNY

WebAssembly and WebCrypto

How they do and don’t help

TOZNY

WebAssembly

TOZNY

- What it is
  - It's assembly language for the browser
  - So you can code browser stuff in something besides JS
- How it helps:
  - Certified or trusted libraries might already be written in another language
  - We use compiled libSodium for compatibility across infrastructure
  - JavaScript crypto might be too slow for some uses (e.g. key derivation)
- How it doesn't help
  - It doesn't solve any of these vulnerabilities
  - You are still shipping the encryption protocol and primitives to the browser

WebCrypto

- What it is
  - Implementations of common encryption protocols by the browser
  - Gives you an API to call out to things like SHA2 and AES
- How it helps
  - We use it for cryptographically secure randomness
  - It can be fast, even take advantage of hardware acceleration for e.g. AES
  - You don’t need to ship encryption primitives in your JS
- How it doesn’t help
  - It doesn’t solve any of these vulnerabilities
  - You still have to implement the protocol in JS, which can be undermined

TOZNY

Bottom Line: WebAssembly and WebCrypto

- They are good!
- They might help us use crypto more by making it available
- They will speed it up by making it native
- But they don’t fundamentally change the security of browser crypto

TOZNY

What does help? Modern JavaScript Security

- HTTP Strict Transport Security (HSTS)
  - Always load this page over HTTPS; prevents downgrade attacks
- Strict Content Security Policies (CSP)
  - Whitelist safe sources for loading code
- Subresource integrity (SRI)
  - Only load scripts that you know you can trust using hashes

TOZNY

Conclusion: Do more application-layer encryption

- It creates defense-in-depth
- It improves privacy, in some cases substantially
- But it's harder for developers than just implementing HTTPS
- A couple pieces of advice:
  - libSodium is easier than picking up raw AES, but it's not NIST approved
  - Easy to use encryption is what we do, so look us up

See my other talks or my "developer's guide to encryption"
for more practical advice on implementing correct encryption
https://tozny.com/blog/encryption-for-developers/

TOZNY

Does Privacy Matter? Apple Thinks So.

TOZNY

Does Privacy Matter? What do *You* Think?

- Do you **do** anything that someone would disapprove of?
- Do you **believe** anything that someone would disagree with?
- Do you **have** anything that someone would want?
- Do you **say** anything that someone would fight against?
- **Are** you anything that someone would hate?

Yes. Privacy matters.

TOZNY

Thank You!

Isaac Potoczny-Jones
ijones@tozny.com
@SyntaxPolice

TOZNY