Whitepaper - A Digital Lock System

Bryan Woods, 2017

Summary

We propose an access-control approach useful for physical and virtual assets. The approach removes the need for physical tokens and shared secrets, replacing them with cryptographic keys held on personal smart devices. (Sections 1 - 2)

We describe a simple protocol that enables any/all cooperating independent implementations to perform access control operations in a consistent and convenient way. (Section 3)

We discuss our open source proof-of-concept implementation that verifies the approach. We provide a link to the source code and documentation published separately. (Section 4)

We present some examples to show how the approach may be applied to simplify access control and identity management, reduce capital and operational costs, and improve security. (Section 5)

1 Introduction

Consider a mechanical lock that accepts a physical key, or “token”. Anyone holding a key with cuts that match the configuration of the lock’s tumblers is able to open the lock. The magnetic card used in many office access systems is another kind of token that uses magnetic fields to transmit credential information (a serial number) to a card reader controlling a door. Whatever the type of token used, it represents a *shared secret* that must be stored and protected by both key holder and lock holder.

Next consider the virtual case in which someone tries to log in to a computer or network resource such as a web site. Our unknown user presents credentials (usually an account name and password) that are compared to values in a table, and if the credentials are found and match then the user is granted access. Login credentials are another kind of shared secret, in this case data stored and protected by both the user and site.

Tokens and credentials have one fundamental security weakness in common: they can be shared, stolen or copied and therefore cease to be secret. They can simply be presented to impersonate another user and gain access without permission.

The costs associated with physical tokens such as magnetic keycards can be substantial. Tokens must be procured, distributed and collected. When lost they must be replaced, reissued and reconfigured. The virtual world of user login credentials to systems and network properties has similar concerns in user management.

Some organizations such as banks bear the expense of maintaining multiple, separate user identity systems for physical tokens (ATM/credit cards) and virtual credentials for online banking. A possible or suspected breach of user credentials could incur the cost of having to reissue a large number of cards and/or update the online account credentials.

Credential theft is a substantial problem. Some estimates claim stolen login credentials were responsible for 63% of all data breaches in 2016 [1]. The impact to the global economy from cybercrime has been estimated at over $400 Billion annually [2], and by year 2020 the total global impact from cybercrime could exceed $2 Trillion [3].

We propose to eliminate use of the shared secret for both the physical and virtual access scenarios. Our approach requires a user to convey a digital credential to a computer-controlled lock by using a personal computing (“smart”) device. Both smart device and lock must perform cryptographic work to communicate and verify the credential. Doing so confirms identity as each credential is unique and managed exclusively by the user on their own personal device.

Our approach removes the need to store secret credentials. Miscreants can no longer steal and use them to access systems by impersonating other users.

2 Background

Digital signature is a mathematical approach to confirm the authenticity of information. It was first described by Diffie and Hellman in 1976 [4] and implemented by Rivest, Shamir and Adleman in 1978 [5] as part of their key-pair cryptography system. Their implementation is known as RSA and remains in common use.

Other cryptographic schemes have since been created including Elliptic Curve Cryptography (ECC) introduced by Koblitz [6] and independently by Miller [7] in the 1980's. The Elliptic Curve Digital Signature Algorithm (ECDSA) uses ECC for creating and verifying digital signatures.

The RSA, ECC and other *key-pair cryptography* schemes require communicating entities to independently generate *key pairs*. A *key* is a number, and a key pair consists of two mathematically-related numbers. One key of the pair is deemed *public* and can be shared openly. The other key of the pair is *private* and must always be kept secure by the holder.

The digital signature of a message is generated by digitally scrambling ("*hashing*") the message data and encrypting the resulting hash with the key holder's *private* key. Only the key holder can sign a message as only they retain the private key.

Any receiver knowing the sender's public key may apply a corresponding mathematical operation upon the signature to *verify* it, that is to confirm that the message was in fact signed using the private key matching the sender's public key. Any alteration of the message will cause the verification to fail and the receiver will know to disregard the message.

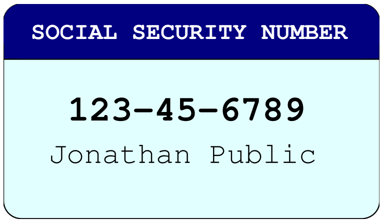
The private key is used by the signer but is neither included within the signature nor transmitted. It is not possible to determine the sender's private key from any combination of public key, message and/or signature. It is also not possible to alter a message and generate a corresponding valid signature of the sender without the sender's private key.

Digital signatures provide a way to mathematically prove that a message came from a specific source and that the message has not been altered. If public keys are shared *a priori* by some mechanism and private keys are kept private then cryptography can be useful for identity management.

In the next section we clarify how a public key can be an identifier and generating a corresponding signature is a mechanism to prove posession of that identifier, hence proving identity.

Illustration by Example

Consider a hypothetical country's social security program. When someone first begins earning income and must pay taxes, they are required to obtain a government-issued identification number for submitting annual tax returns and someday claiming a retirement pension.



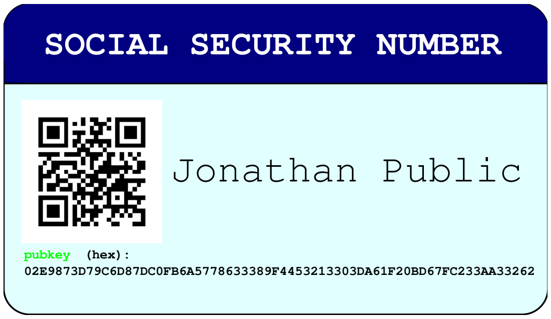
SOCIAL SECURITY NUMBER

123-45-6789

Jonathan Public

The number is simply a sequentially-allocated serial number that has no other meaning. It is as if the card number is an account name and the name is an unencrypted password. As long as the information is kept secure by person and agencies the process works well enough. However, any person who somehow learns the number/name combination could use the information to file false tax returns and fraudulently receive benefits.

Now consider the advantage of a similar system in which one can *mathematically* prove ownership of that social security account by responding to a digital challenge. When someone first applies for a social security account, they create a cryptographic key pair and provide the public key to the agency. The agency simply records the public key's value and uses that as a primary index for all related records.



SOCIAL SECURITY NUMBER

Jonathan Public

pubkey (hex):
02E9873D79C6D87DC0FB6A5778633389F4453213303DA61F20BD67FC233AA33262

In the future our person requests service from the social security or taxation agency. The agency presents a new challenge and our person identifies by providing a public key and a cryptographic signature confirming ownwership of the corresponding private key. The agency then retrieves our person's records using the person's public key.

The message exchange does not reveal the private key. If a third party listened on the message exchange they may at best learn a person's public key, but not the private key. Without the private key they cannot create the signature for a specified public key, and therefore cannot impersonate someone else.

3 Approach

Notation:

(K,k) is a key pair in which k is a private key and K is the matching public key
SIG(m, k) is a digital signature of a message m generated using private key k

3.1 Challenge

Traditionally a user approaches a lock, presents a key and attempts to open (“tries”) the lock. In the virtual world a user approaches a computer or web site and tries the lock by submitting a username and password.

Our system takes the opposite approach: the lock opens the conversation by presenting a challenge openly to the world and then waiting for a response.

The lock begins by generating a new random key pair (G, g). We use the public key G as the message to be signed thus SIG(m, k) is actually SIG (G, g). The challenge message is a simple concatenation of the byte values for the two fields:

G SIG(G, g)

We use the elliptic curve known as *secp256k1* to generate key pairs and ECDSA for signatures. The choices were motivated by the following:

- ECC provides more cryptographic strength with smaller key sizes than other schemes, and we need to keep message sizes to a minimum due to technology limitations.

- Curve secp256k1 and ECDSA are components of the well-known Bitcoin system as well as other digital currencies.

- ECC and ECDSA code libraries are available as open-source. These libraries have been well tested and have support from the cryptographic community.

The lock must generate a new G on startup and every time the lock receives a response. This is critical to avoid replay attacks in which a successful response is observed and somehow recorded, and then presented again later.

3.2 Response

Having issued a challenge, the lock awaits a reply to the challenge. When one is received, the lock parses the response to verify the signature. If x is the signature part of the challenge such that x = SIG(G, g), then the format of a valid response[a] is:

K SIG(x, k)

The response is simply the concatenation of the user's chosen public key K and the signature of the *challenge signature*. Note the user signs the challenge's signature, not his own public key, K. This distinction is important as it forms a mathematical relationship between the response to the challenge.

The lock receives the response and parses the K and signature fields. It then confirms that the message signed by K was the outstanding challenge's signature field, x.

Now the lock knows that the responder controls public key K. Our approach requires authorized users to register public keys with locks by some external process. The lock side maintains an authorized-keys list from which it can search to locate a match for a given value of K.

This challenge response exchange is suitable for direct user access in situations where the user and the lock are physically colocated. The exchange and protocol is illustrated in Illustration 1.

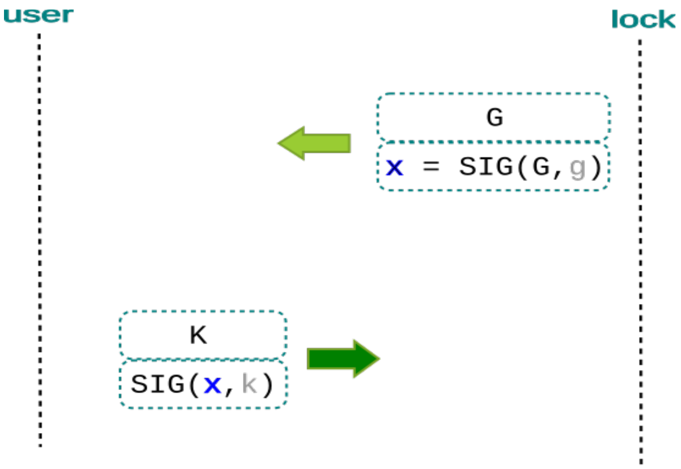


Illustration 1: Challenge/Response Exchange (Physical)

[a] Our protocol actually defines *two* response types. This one is only suitable for the physical case in which a user and the lock communicate directly, without any network or intermediary. The other response type is used for access to virtual resources.

3.3 Agent

Now consider the virtual case, in which a user holding a personal smart device is at a computer (e.g. desktop) and tries a remote lock on the network. We introduce a software *agent* to act as an intermediary between the user and the virtual lock.

Our approach requires the user to retain possession of his/her private key(s) on the personal device. The user never exposes his private key to any other computer.

The agent interacts with the lock on behalf of the user. The agent cannot sign as the user because it does not have the user's private key.

We extend the protocol to include a *session* in which all parties are aware of one another's existence, and confirm the user has granted permission for the agent to act as intermediary.

The exchange proceeds as follows (refer to the interaction diagram in Illustration 2):

i. A user at a computer runs an agent and attempts to try a remote lock. The agent generates a new keypair (A, a) as it does for every new session.

ii. The lock generates a new random keypair (G, g) for every new connection. When the agent connects, the lock generates a challenge just as in the physical case and returns the (key, sig) combination to the agent:

   G SIG(G, g)

iii. The agent examines the signature field:

   x = SIG(G, g)

   The agent verifies G's signature and then signs x using private key a. It then sends a two-part message to the user, where each part is one (key,sig) combination:

   G x A SIG(x,a)

iv. The user notices the presence of two message parts and realizes an agent is present. The user verifies the signature chain, then uses the agent's signature:

   y = SIG(x,a)

   The user generates SIG(y, k) and returns a one-part message to the agent:

   K SIG(y, k)

v. The agent verifies the signature:

z = SIG(y, k)

The agent has retained the lock’s challenge and now assembles a three-part response to send to the lock:

G x A y K z

vi. The lock verifies the three signatures and confirms G is an outstanding challenge.The message signatures in the responses are “chained together”, that is the agent signed the lock’s signature and the user signed the agent’s signature. This establishes a logical *session* including user, agent and lock. The user received a two-part challenge and knows the lock and agent. The lock received a three-part response and knows the agent and user. The lock can trust the agent and return data on the connection for the user.

vii. The user’s key K is checked against the access control list. If found, the session is established (the lock is “open”) and communications can occur until the connection is closed.

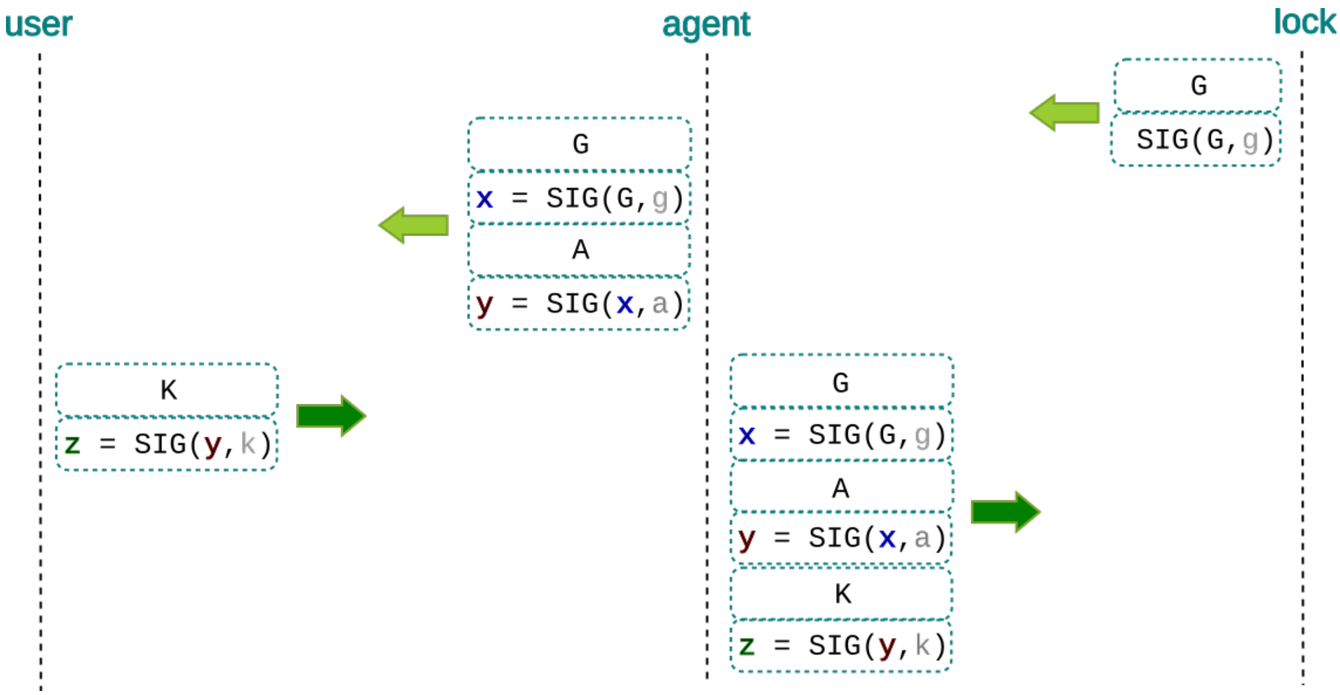


Illustration 2: User/Agent/Lock Interaction

4 Implementation

Our test/reference implementation system is called **ADILOS** (**A DI**gitial **LO**ck **S**ystem) [8] and includes these subsystems:

keymaster: [9] an application for the user's personal smart device

gatekeeper: [10] an example of a software-controlled physical lock

kgagent: [11] a software program acting as an agent between keymaster and kgserver

kgserver: [12] a simplified web server acting as an example of a 'virtual lock'

The keymaster maintains a set of named cryptographic keys. It enables the user to create keys and respond to any challenge using a selected key. It can show a public key for sharing with another party and reveal a private key for importing into another wallet.

For the physical scenario, a gatekeeper creates a challenge by generating and displaying a "Quick Read" (QR) code. A user runs keymaster and scans the challenge QR code using the smart device's high-definition digital camera. The keymaster computes the response including the user's selected key and displays the response as a QR code on the device screen. The user holds the smart device's screen in front of the gatekeeper's camera. Note that communication is entirely optical; no network connection is required.

For the virtual scenario, the user begins by running the kgagent program. This agent is a simplified web browser that can fetch and render HTML files retrieved from world-wide web sites using HTTP. Additionally, our implementation supports a "kg:" protocol to instruct the agent to connect with kgservers. When the user provides a "kg:" locator (instead of "http:" or "https:"), the program connects to the kgserver over the network.

The kgserver answers the new connection by returning a new challenge to the agent. Our agent receives that challenge and forms the two-part message for the user displayed on screen as a QR code. The user runs keymaster and scans this code, approves the agent by signing and generates an encoded response. This response is provided to the agent which then forms the final response for the server.

The aim of our default implementation is to test and validate our approach. It is released as open source to facilitate independent verification and encourage adoption of the protocol. Any person or organization may create alternate implementations specific to their needs. If all use the same protocol, any keymaster can interact with all agents, servers and gatekeepers.

The code may also serve as an open, extensible resource to facilitate future development and integrations.

5 Application Examples

An office could replace the magnetic card reader at the entrance with a customized gatekeeper and avoid the expense of producing and maintaining the inventory of cards. Whenever an employee is hired, the hiring manager asks for a public key. The new employee creates a new key on his own smartphone and displays it for the manager to scan into the office management system. Our new employee uses this same key to enter the office, login to the computer network, and access the corporate fitness centre.

A larger and/or distributed organization such as a university campus with many access systems could deploy customized gatekeepers that read one common authorized keys list maintained at by facilities management at one location. This would provide one place to manage access to laboratories, equipment storage, dormatories, library, sport facilitiies, the student lounge and so on. If the university used public keys as student numbers then the same key would be used for student records.

A club could use ADILOS to avoid having to maintain a collection of physical keys to clubhouses and shared assets such as storage facilities.

A parking garage could implement a gatekeeper and thereby simplify access for patrons who rent by the month.

A web site operator could use ADILOS to remove the need for two-factor-authentication (2FA) support. This could save money by removing email pumps, SMS relays and shifting number-display tokens. Removing these mechanisms lifts a burden on users as well.

A hospital could implement gatekeepers for narcotics supply cabinets to centralize access control and logging in one place. This centralized datastore would also facilitate audits and investigations. Police organizations could implement ADILOS to control evidence lockups for similar reasons.

A hotel could put a gatekeeper on the door of every room. The guests would show a public key at checkin, the hotel would scan and tie that key to a room. When the guest approaches the room he uses ADILOS to confirm his key. The gatekeeper on the door queries the hotel’s computer system to see he is allowed access. This would save hotels the cost of maintaining and programming the room access keycards.

The air travel industry could simplify traveller engagement by using ADILOS to track the traveler from ticket purchase, through security and to the gate. Whenever required, our traveler uses his smart phone to verify his public key. This would remove the burden of holding paper tickets, boarding passes, passports and travel/work visas. Ultimately travelers could go anywhere in the world without a wallet, simply using their phone.

A bank could eliminate the costs of issuing ATM/debit/credit cards and reissuing those cards whenever there has been a possible or suspected credential theft. The customer could provide a public key when first opening an account. This same key could be used for point-of-sale, ATM access, identification at service counters and for logging in to online banking. The ATM hardware would be less expensive to purchase as there is no longer a need for PIN pads and card readers. Operational costs would be reduced as there are fewer points of attack (e.g. card skimmers) and no opportunity for fraud by credential theft.

A state government could reduce the cost of providing identification services by no longer producing physical drivers licenses. A new driver could register a public key on the Department of Motor Vehicles web site when taking the test. If passed, the DMV would simply update a record in its database. If that driver were stopped by the police he could provide proof of identity and valid possession of a licence by using ADILOS to respond to the officer’s digital challenge. The officer would obtain the driver’s public key and use that for the lookup in the DMV system.

ADILOS could reduce costs for event management and ticketing organizations by replacing paper tickets. This could eliminate counterfeiting and scalping. Someone attending a sports event could use their ADILOS key to pay for the ticket with bitcoin. There would be no need for a receipt because the transaction is publicly viewable on the blockchain. At the venue, the customer would use the same key to respond to a challenge at the gate. The specialized gatekeeper would get the public key and convert that to a bitcoin address. The gatekeeper would then checks the blockchain to confirm payment was received from that address, find the transaction and open the gate.

ADILOS could free users from physical keys, magnetic/RFID cards, bar-coded swipe badges, bank cards, identification cards and numerous computer logins. The keys could be stored within an application and can be backed up manually or with a service provider’s cloud backup mechanism. If a device is lost it could not be used by others due to PIN protection at the device and app level. Once a new device was obtained the app could be reinstalled and restored from backup.

With ADILOS the user chooses how to manage keys. Many users might be content to simply use one key for everything. Others may wish to segregate keys by application, such as one key for government identification, another key for banking and a third key for everything else.

6 Conclusion

Our system uses public key cryptography and supporting technology to present a simple, open approach to access control for physical and virtual assets. The approach is also useful as an identity management system.

The approach eliminates stored secrets and therefore provides higher security for users and organizations. The opportunity for fraud by stealing tokens or credentials from sites is removed. This approach reduces or eliminates significant future losses due to cyber-crime.

The ADILOS protocol enables any party to make specialized locks or alternative key-management apps and enjoy the ability to have their product work with all other products. Our default implementation serves as illustration of the protocol and as a reusable component to help adopters integrate.

ADILOS presents an opportunity for cost reduction by reducing hardware requirements and simplifying operations for identification and access control.

ADILOS obsoletes physical tokens, names+passwords, two-factor authentication mechanisms and identity schemes. ADILOS replaces those with a simple approach that:

- uses standard, well-known and well-tested cryptographic algorithms,
- is open-source, inspectable, extensible, and aproprietary,
- works without cell, wifi or any network service,
- is independent of central authority or third-party support,
- maintains privacy between user and lock holder, and
- places control of identity with the individual.

A References

[1] Verizon Data Breach Investigation Report http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/?utm_source=pr&utm_medium=pr&utm_campaign=dbir2016 (warning: paywall)

[2] http://www.telegraph.co.uk/technology/internet-security/10886640/Cyber-crime-costs-global-economy-445-bn-annually.html

[3] https://www.juniperresearch.com/press/press-releases/cybercrime-cost-businesses-over-2trillion

[4] "New Directions in Cryptography", IEEE Transactions on Information Theory, IT-22(6):644–654, Nov. 1976

[5] *Rivest, R.; A. Shamir; L. Adleman (1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" (PDF). Communications of the ACM.* ***21*** *(2): 120–126.*

[6] *Koblitz, N. (1987). "Elliptic curve cryptosystems". Mathematics of Computation.* ***48*** *(177): 203–209*

[7] *Miller, V. (1985). "Use of elliptic curves in cryptography". CRYPTO. Lecture Notes in Computer Science.* ***85****: 417–426*

[8] https://github.com/bitsanity/ADILOS

[9] https://github.com/bitsanity/keymaster

[10] https://github.com/bitsanity/gatekeeper

[11] https://github.com/bitsanity/kgagent

[12] https://github.com/bitsanity/kgserver