

Trusted Identity of Physicians in Cyberspace Hearing

July 11, 2012

9:00 a.m. to 3:00 p.m./Eastern Time
The Dupont Circle Hotel, Dupont Ballroom
1500 New Hampshire Avenue, NW
Washington DC 20036

Steve Kirsch, CTO, OneID

900 Island Drive STE 204
Redwood City, CA 94065
stk@oneid.com
650-279-1008

What is OneID?

Novel approach to identity

Designed from scratch to be NSTIC compliant:

- Easy to use: OneID is as easy for consumers use as Facebook Connect. It is as easy for developers to deploy (in as little as 30 minutes).
- Secure: the OneID identity repository (which holds user attribute information as well as some crypto secrets) is fully encrypted using keys held on user's endpoint devices making it impossible for the identity repository provider (OneID in most cases but it could be GSA, HHS, etc) to decrypt the user's data. This security is verifiable because unlike most every other identity provider (IdP), the code dealing with a user's identity is publicly inspectable at all times since it the code that protects the users crypto secrets runs in the user's devices and not behind a firewall in a cloud server. Therefore, there is no need to trust the identity provider policies. The architecture of the system and public inspection and verification of the code running in the user's devices guarantees the security of the data and the integrity of the system no matter what code is running in the cloud server.
- Private: the relying party doesn't know your IdP, and your OneID cloud identity repository provider doesn't know anything about what you are doing. In addition, relying parties (RPs) cannot track you because OneID generates unique public keys for each RP (i.e., opaque identifiers are used).
- User centric: Unlike most other IdP architectures, in OneID, only the user himself can assert his identity. This is because required crypto secrets to assert an identity are only stored on the user's endpoint devices. So, for example, today, facebook or Google can easily lie (due to a bug or an attack) and claim the user "Steve" authenticated into their system. With OneID, this is not possible because authentication is end-to-end secure: the user device signs a challenge supplied by the RP and the signature is verified at the RP. The assertions of middlemen do not play a role.

- Multi-provider: A user can choose which OneID-compliant identity provider he wants to use to store his identity; they are not forced to store their identity in OneID's repository.

Public launch October 2012

OneID is available now for software developers.

What does OneID enable?

Authentication (AuthN)

- Trusted high-assurance authentication using devices you already have. Supports LoA up to NIST 800-63 Level 3 with no special hardware and can support Level 4 with a smartcard (such as a PIV, CAC, or NXP SmartMX2 smartcard).
- Authentication can be done both on-line (e.g., on websites) as well as off-line (in-person and over the phone).
- Authorization (AuthZ)
 - End-to-end secure transactions. For example, a prescription can be digitally signed by a doctor and verified by the pharmacy that fills the prescription. No system in between can tamper with the prescription. In fact, they can't even read the prescription if the doctor doesn't want them to.

Digital claims

- You can prove: "I have a license from California Medical Board." This is done by asserting the license certificate to an RP and signing a challenge (or a prescription) using the private keys associated with the public keys in the certificate. In practice, this is done with a single click on a website. The doctor doesn't even know that public keys are being used.
- OneID also allows you to prove things without revealing details, e.g., you can prove "I am over 21" without revealing your birthday
- Authorization can be easily revoked with a certificate revocation list maintained by the authority issuing the certificate.
- OneID supports RP-created attribute sets similar to Microsoft Infocards. So a physician can accept things like single or multiple medical licenses, passports, driver's licenses, etc.

Info storage/sharing

- Public keys, attributes, digital certificates, proof of purchase, RTU licenses, etc. can be stored in a OneID identity.
- OneID supports Auto-fill on web pages if the RP has suitably coded the page to map the OneID attributes into the form fill.

What's unique about OneID?

- A single identity usable for web, desktop, mobile, and enterprise apps. It is usable off-line and on-line
- Identity can only be asserted with express consent of user
- Mass consumer adoption (many doctors will already have it)
- Paradigm shift: shared secrets -> 3 independent public key pairs
- User friendly: crypto management is hidden from users, two-factor, OOB/PIN LoA
- NIST 800-63 Level 3: Uses NSA Suite B crypto (ECC P-256)
- Secrets are distributed (including user endpoint devices)
- The architecture (not operational policy) guarantees a mass breach is impossible @ RP, IdP (no more LinkedIn type breaches)
- End-to-end secure transactions: signed at the producer, verified at the consumer of the transaction
- It's free
- Device centric: you must "pair" your identity with new devices before use
- Immune to all common identity attacks: phishing, key logging, malware, MITM, MITB, etc. OneID requires an attacker compromise 6 secrets (2 independent crypto keys on 2 different device classes and a password and PIN), but the best attack can only obtain 4.

A top architect for a large US government agency remarked after a 3 hour briefing, "This is exactly what the government needs."

OneID can be used for off-line authentication and authorization

OneID is applicable to on-line, in-person, and over the phone authentication and authorization.

For example, a physician can prove to a pharmacist over the phone that he is really the Dr. Eric Weiss of Woodside who wrote the prescription. Dr. Weiss starts up OneID, enters his PIN code (which is asymmetrically verified), gets a 3 digit code, tells the pharmacy that three digit code, the pharmacy enters the code into their system, and Dr. Weiss then confirms it in his mobile app. The pharmacy software then compares the signature obtained with the public keys on file for the doctor (in their database, on the prescription, or obtainable on the web), and the pharmacy is guaranteed of his identity. It looks like this:

- Dr. Weiss: Starts app which displays "123". Tells pharmacy.
- Pharmacy: <types in 123 into their system>
- Dr. Weiss: Sees in the app that the pharmacy wants to confirm his identity. Hits "OK"
- At this point the pharmacy has authenticated the doctor.

OneID auth mimics real life

The way OneID works is not that much different from how we do things today.

- **Physician:** “Hi. I’m Dr. Fred Smith. Here are my credentials issued by the California Medical Board. Here is my signature.”
- **Relying Party:** “Your signature matches the one on the cert. Let me next check to make sure your cert wasn’t revoked....OK, you’re authenticated.”

Result: higher security, greater ease of use because the doctor needs only remember his OneID password.

Issuance of physician credentials (example)

Physician logs into medical board where he has his license, e.g., mbc.ca.gov and clicks a button on the page: “Add license cert to my OneID.”

The License certificate is added to the physician’s identity is now usable on all the devices used by the physician.

OneID also allows a great deal of flexibility in the choice for the public keys in the license certificate:

- Doctor GUID (greater security since the RP can more directly verify attributes associated with other certificates the doctor may have). GUID= “global unique identifier”.
- mbc.ca.gov specific (preserves privacy)
- Doctor supplied UID (preserves privacy)

So if privacy is an issue, the public key set can be unique either to the medical board, or chosen by the physician. For greater identity assurance, the public key set can match the public key set associated with the physician’s true identity. We recommend the latter since this allows more things to be tied together, e.g., California requires Live Scan Fingerprinting which can be tied to the GUID of the doctor.

Acceptance of physician credentials at a pharmacy (example)

Doctor hits “Authorize prescription” button in his electronic health record (EHR) system, e.g., eClinicalWorks (eCW).

This causes the EHR to request the browser (or OneID desktop or OneID mobile app depending on the EHR client type) to digitally sign the prescription with the doctor’s signature and include his license. The electronic prescription should include the name of the pharmacy (or chain) to prevent duplicate fills.

Prescriptions can be pure digital or they can be printed out in the form of a short QR code that contains a short URI pointing to the prescription. Ideally, they are encrypted so that only the authorized pharmacy (or other entities) can read the prescription.

Since only the pharmacy can read the prescription, the pharmacy must verify the doctor's signatures and certificates on the prescription. Either the pharmacy can write its own software, or deploy a 3rd party verification service to decrypt and validate the prescription.

The prescription might even contain the OneID identity of the patient which can include his address and biometric information. In the case of in-person pickup, the patient simply presents his index finger to a biometric scanner to pick up the prescription.

Suggested requirements for a secure identity system

The suggested requirements below are desirable characteristics for a secure identity system. They are also achievable because OneID can meet all of them.

1. Pick an secure identity supporting two-factor, out-of-band digital signatures that doctors already have (or will soon have). It would be bad to replace username/passwords with having to manage multiple identity systems! Pick the identity system that has the most potential to be a secure identity system standard.
2. Simple APIs that require only a few lines of code to implement.
3. The identity system should not have any shared secrets, i.e., secrets that are shared between two different parties. Examples of shared secrets are passwords, credit cards. To avoid share secrets, authentication and authorization should be done by asymmetric cryptography, e.g., ECC.
4. Avoid SAML. It's confusing, hard to understand, hard to implement, underspecified, and hard to configure. Vendors like it for that reason. There are better, more modern and more secure options available.
5. Avoid OpenID. It was designed to aggregate traditional IdPs. The IdP makes a representation directly to the RP which is problematic for two reasons: 1) the IdP knows the identity of the RP (a privacy violation) and 2) the IdP can lie so mass breaches are easy.
6. The system must use NSA Suite B crypto such as ECC P-256 and AES-256.
7. Demand end-to-end security. Trusting middlemen (such as IdPs that assert a user's identity instead of the user device itself) is always a mistake. Would you trust LinkedIn to assert someone's identity? Not only was LinkedIn a victim of password theft (all the passwords were stolen), but it is also vulnerable to making false assertions.
8. Secrets must be held on user endpoint devices. You must use an identity system that provisions secrets held on endpoints and it is those secrets that are required for signing RP challenges.
9. Require at least 2 signatures to authenticate @ an RP for a high assurance transaction. One signature is not enough because any single device can be compromised. Never trust a middleman IdP who claims to have verified the signatures.
10. Immunity to mass identity breaches should be guaranteed by the system architecture, not the operational policies and procedures of the IdP. We have the technology now so you no longer have to trust people and operational policy. Put your trust in the algorithms and the architecture. Look at PCI compliance for example. We have proven time and time again that it can be breached because it relies on process assurance. A secure system would eliminate the

shared secrets, replacing credit card information (the shared secret) with digital signatures. Breaches would be impossible and there wouldn't have to be any PCI regulations at all. You have the opportunity to specify it correctly here.

11. Users must be able to move crypto secrets easily and securely between devices. They should not need a PhD from MIT to do this. This should be done using a cloud pairing server, but the architecture must guarantee that the pairing server, if compromised, cannot sniff any of the secrets.
12. No single point of compromise (SPOC). The system integrity should not be compromised if any single component or entity is breached. Any attack should, at worst, stop validation of an identity; it should never allow an attacker to assert an identity.
13. Account recovery without a high entropy secret supplied by the user must not be possible. There are too many systems out there that will "reset your password" if you answer some KBA questions or verify your email. Those systems are catastrophes waiting to happen. Anytime a system allows a low entropy reset, it means that the IdP is able to assert anyone's identity at any time with a simple code change. The only way a user should be able to recover his identity is through use of a high entropy (256 bit or more) secret.
14. User should be able to pick IdP provider (i.e., where their encrypted data is stored in the cloud).
15. Identity cannot be asserted without express consent of the user. So even if the IdP is breached, the attacker should not be able to instantly assert anyone's identity.
16. Identity should support 2-factor + out of band (OOB) authorization using devices a user already has. RSA SecurID is 2-factor, but it is in-band (put into the same device you are accessing) and is susceptible to all sorts of attacks such as MITM, MITB. In addition, hardware tokens such as SecurID, YubiKey, etc. do not "sign" the transaction itself and therefore are much less secure than OneID which provides out-of-band verification of the request and digital signature of the transaction itself.
17. The identity should be immune to phishing, key logging, malware, MITM, MITB, and physical theft of all devices attacks
18. Personally identifiable information (PII) should never be stored in a way that is retrievable by any attacker.
19. The system should allow for easy provisioning @ authority, e.g., it should as simple as a single mouse click to "pickup" a medical board certification and add it to your digital identity.
20. Users should be able to self-manage their own devices, e.g., if a device is lost, it should easy to disable the lost device and re-provision a new device.

Final thoughts

Start modestly and simply with a simple pilot involving just a few parties.

Licensing authorities need only to modify their website to enable certificate provisioning, and then publish a single file on an FTP server: the certificate revocation list (CRL).

Relying parties at both ends of the transaction (EMR system vendors, ePrescription companies, pharmacies, etc.) need only add a small amount of code to accept a digital identity and to validate the identity.

It is probably best to leave existing middleman infrastructure (e.g., Surescripts) intact but add the security at the endpoints (doctor office and pharmacy) so that the entire system is end-to-end secure. This can be done by adding an additional field to existing prescriptions: the electronic signature of the physician is added to the prescription by the EMR software, and the signature is verified by the pharmacy that fills the prescription.

My suggestion is to do a pilot with just three parties:

1. A large Surescripts Gold Solution Provider (either of EMR software or standalone e-prescribing software) that uses a web interface
2. A large pharmacy chain or large on-line pharmacy (e.g., Express Scripts)
3. A trusted identity provider who can meet the suggested requirements above