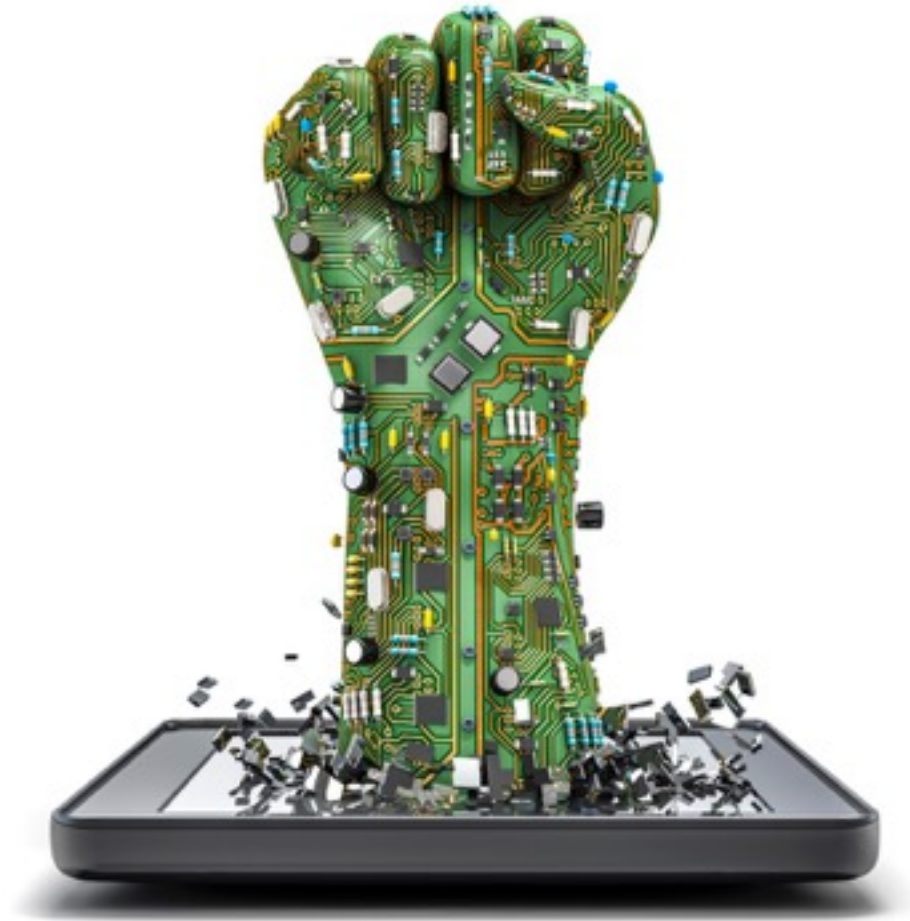


Solving the Digital Security Paradox



**“How an app-HSM hybrid revolutionizes the world of
digital security”**



Ubiqu – December 2015

Summary

"Ubiqu and the key store app: A story on how the concept of an app-HSM hybrid revolutionized the world of digital security."

This white paper outlines the dire state of computer science and security, what the problems are and how the revolutionary app-HSM Hybrid fixes them.

One might think the limited scalability and lack of flexibility of hardware secure elements in combination with Public Key Infrastructure (PKI), such as chip cards, is a big problem, but it's even worse: in a connected world with an attacker controlling the end-point the same attacker will also necessarily control the related chip card and codes. On the other hand all other alternatives, i.e. lower level security solutions based on software, lack credible security guarantees. Thus there is no currently available solution for medium and high-risk transactions and login as evidenced by all newspaper articles on high profile hacks.

The underlying problem is:

With every end-point containing bugs and vulnerabilities, compromises at some point are inevitable. This results in malware infections both PCs and phones. The traditional trust anchor used for security, in the form of a secure element, cannot be relied upon anymore, because malware is located between it and the user and can freely abuse it. The result is that phishing, man-in-the-browser and skimming are very familiar concepts to all of us.

A new approach is introduced by Ubiqu, which will once again bring chip cards and secure elements to the forefront of security by combining a chip card with the best of software and by adding something 'extra'. It brings back the chip card level security even on compromised end-points with the concept of an app-cloud hybrid solution. This app-HSM Hybrid is marketed as "Ubiqu" by Ubiqu and experienced by the end-user as the Ubiqu Key store app or in-company-app-solution.



Such a software and ‘hardware-as-a-hybrid’ concept brings the unique feature that:

“It creates a secure trust anchor to any device.”

The app-HSM trust anchor can be used even in compromised environments. Enabling banks, governments and military to secure the high-risk transactions, while also enabling all other organizations to secure their low and medium-risk online businesses without neither Public Key Infrastructures (PKI) and chip card level cost nor complexity. Thus, truly ushering in a secure digital world.





White paper

At the dawn of computers all was relatively well. One central computer catered to many different dumb terminals, but even then a hacker could gain access to the central computer and impersonate any and every user. PKI was invented by RSA, allowing the central computer to verify their users' identity. Combined with a chip card from the Telecoms the problem of impersonation was now solved.

Unfortunately, chip cards and PKI proved to be complex, but more importantly, lacked flexibility and scaled very poorly. Thus only banks, governments and the military adopted them. Those were the only organizations which had a positive business case given the high cost. The rest of the world accepted the risks associated with bad software and servers getting hacked, and turned to detection and accepted loss. This status quo was fine until the transition from the era of the personal computer to the Internet era.

“ When all end-points went online, everything collapsed... ”

Malware, phishing and viruses took over the end-points and banks, governments and the military lost their trust anchor and the trust of their users.

Losing their trust anchor for the digital world meant massive financial fraud from online phishing for banks, the loss of intellectual property for enterprises, the loss of sensitive data for the military. The digitization of governments was ground to a halt.

In this day and age, when a system administrator of a highly sensitive system, using a chip card, puts his chip card in his PC, the following might happen:

If malware is present on his PC, this malware can now use the chip card unnoticed and uninterrupted to gain access to each and any system this administrator has access to, even systems he did not intend to access in that session.

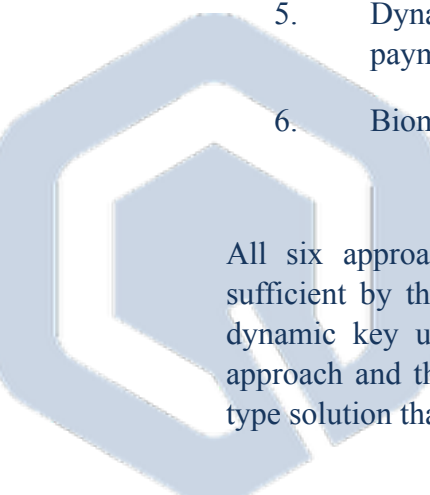
When bank customers transfer money, these customers have no idea that malware has changed their transactions, unintentionally authorizing the transfer of large sums of money to foreign bank accounts.



Software by definition contains bugs thus malware and access to central servers systems cannot be prevented. This results in fraudulent transactions, loss of information and many other infringements. Nevertheless, in the case of Ubiq's solutions the impact is contained and several mitigation techniques, which are explained further on in the paper, marginalize the associated risk. Yet, technology marches on: More digitization, remote cloud computing, more online persistence, Internet of Things and everything mobile, and thus more opportunities for malicious abuse. Staying ahead of the curve and using new approaches is not only needed, but required.

At this moment there are six basic approaches being used and experimented with:

1. Secure elements;
2. Out of band tokens, such as SMS, phone, OTP and mobile apps;
3. Secure Software, employing obfuscation, integrity checks etc.;
4. Device binding;
5. Dynamic key update, in the form of host card emulation for contactless payment on Android mobile phones;
6. Biometrics.



All six approaches have inherent weaknesses and none of the approaches are sufficient by themselves to create a secure solution, although secure elements and dynamic key updates are necessary at a minimum. Below we will discuss each approach and their inherent weaknesses and we will end with the app HSM hybrid type solution that enables a strong security claim on the same level of a chip card.

1. Secure elements

A secure element is the generic term for a certified chip card. It entails a purpose built chip with the sole function of generating, storing and using cryptographic keys. Using a Secure element enables the strong claim of high security assurance. This is because a secure element is a confined and limited environment, where it is possible to minimize the risks of bugs and mistakes and by minimizing the impact of bugs and errors.

Typically the security life cycle of a secure element is 3 years, after which some form of technological advance results in a vulnerability to that generation of secure elements and requires the need to upgrade them to a new generation. This is widely seen e.g. in the banking industry with token issuance and control. Secure elements,



being hardware, have the problem that, once in the field, security vulnerabilities cannot be fixed and thus a new physical secure element needs to be issued.

“ ...Secure elements are useless if malware can come between a user and the secure element... ”

For instance, with modern SIM cards on modern platforms, there is a software firewall implemented on the host OS that determines which app can access which part of a secure element. Once the app and/or the host platform are compromised, the attacker has control over the key material in the secure element.

In general, these risks are brought down to acceptable levels by sending transactional context to the end user, informing them of transactions, and by employing risk engines to filter out unusual transactions.

2. Out of band tokens

Out of band (OOB) tokens are hardware or software tokens where a code is sent to a user using a second channel, for instance an SMS or phone call. Alternative implementations use ‘code generators’ for One-Time Passwords (OTPs) that are time limited.

“ Hardware based OTP combined with a password or biometric entered on the same OTP device can achieve high levels of assurance just like chip cards. Out of band tokens are useless if malware can come between user and the server. ”

When the user enters the OTP codes, there is no connection between the action of the user and the entry of their security codes. Malware can alter the site where one is logging in, and log them in somewhere else completely, or change a payment transaction unnoticed.

As for secure elements, risks are brought down to acceptable levels by sending context to the end user informing them of transactions and by employing risk engines to filter out unusual transactions.



3. Secure Software

Secure software is a hot research topic especially for the mobile platforms.

With secure software we are referring to all tooling intended to add obfuscation, encryption and integrity checks to running software with the intent to prevent hackers from understanding the flow of the program, manipulating the code or learning specific sensitive values.

Mobile platforms are generally regarded as safer than computers, because more security and scrutiny is being employed on these mobile platforms. Banks report almost negligible hacking on the mobile platforms as compared to phishing on desktop platforms. Mobile platforms being safer are not inherently safer than desktop platforms and the current gap will most probably be narrowed or closed in the coming year(s).

“ One fundamental problem that all software based solution face is key storage. ”

The trust anchor ultimately is a cryptographic key, and once an attacker has acquired a copy or trace of a running program, the hacker can analyze it off-line until the secret is extracted, compromising the trust anchor. The inherent problem is that stored keys are static and as such will be valid during the usually long time it takes to do off-line key extraction. A copy of the program can typically be made by malware. The end goal being the retrieval of the (a)symmetric keys. Retrieving these private keys enables a hacker to use them maliciously. The complete infrastructure and apps surrounding the private keys will no longer be necessary, which is a very undesirable property.

Next to risk mitigation strategies associated with secure elements and OOB tokens, secure software risks are also typically mitigated to acceptable levels by limiting the amounts and types of transactions..

4. Device binding

Device binding is a security approach where specific checks are added to the runtime of an app. Device identifying information is collected, aggregated and condensed to a fingerprint, which is compared to a previously stored fingerprint typically collected either during the last use session of the app, or during the registration period.



Comparing these two fingerprints server then provides a way to ensure a specific installation of an app is still running on the same device as the last time it ran or that it was installed on (and thus wasn't copied).

Device binding is useless if malware can intercept the system calls that provide the information that is used to identify the device in the first place, making cloning of apps possible once the underlying OS platform has been compromised. Device binding techniques that don't use system calls, but extract device identifying information themselves are harder to circumvent, but is not always possible on every platform.

5. Dynamic key update

A novel approach is the idea of dynamic key updates. The idea is that the keys in the application don't live long enough for a hacker to hack through the secure software and device binding measures to extract useable keys from the solution This kind of approach is seen with symmetric keys in the context of Host Card Emulation (HCE) and with asymmetric RSA keys in the form of mobile phone apps.

“... When used in this way public or symmetric key distribution to relying parties starts to become an issue and adds a lot of complexity due to the frequent rotation of these keys. The rotation of large asymmetric keys is also time consuming, if proven safe keys (like RSA) are used.”

6. Biometrics

An even more innovative approach is keystroke dynamics or behavioral/continuous authentication, which brings back the promise of direct authentication of the user by way of his behavior of the early 2000's. In general, biometrics have some inherent weaknesses:

1. When the biometric data is acquired locally and compared centrally to a template, privacy and identity theft immediately become a major issue. Who guards and controls the use of the biometric data and templates on the server?
2. When the biometric data is acquired locally and compared locally, the problem is shifted to having a trust anchor to trust the local comparison.



3. Biometrics have a heavy dependence on sensors that can readily be fooled by creative human beings or smart software that can replay the output of the sensors or the input to the sensors.
4. Getting the false positive and false negative into a reliable zone of usefulness without hampering usability over a very broad range of demographics is a real challenge, but is also critically needed for large-scale adaptations.
5. Handling variability of the biometrics by natural and less natural changes in people is also a critical and difficult to tackle issue.
6. An identity cannot be revoked, giving control to assailants and the collectors of the data, instead of the user. This hinges on the fact that there is no replaceability concept in most types of biometrics. If anything gets leaked regarding the biometric templates, the end-user can effectively no longer use that specific biometric, there is no replacement process possible.

“ We claim that the App-HSM hybrid is the only solution currently available that can successfully combine the before mentioned six approaches and also do it on a world population scale. “

Resulting level of security when combining all of the above

A sound approach would be to combine all or most of the above. The combination would allow us to create a tightly interlocking solution, in which one approach can mitigate the risks posed by implementing another approach. We claim that the App-HSM hybrid being presented in this white paper is the only, currently, available solution that can successfully integrate all 6 before mentioned approaches and add the crucial ingredient to make it work.

When combining the above-mentioned solutions naively the following situation arises. Let's assume malware on a mobile device is able to gain root access, but has not been optimized to attack the application. If a secure element is used, the malware can just mangle its way between the app and secure elements. Just as when only secure elements are employed.

If the above scenario is made more sophisticated and a secret is employed to connect an application to the secure element, the malware now only has to have a copy of the app's data and the hacker is able extract the secret offline, just as with the “secure software” case above.



“ Device binding is of no use in this scenario, since it's implicitly bound by there being a secure element on the phone anyway. “

Current best practice

The best currently employed solution is to connect to a secure element on a mobile phone remotely via a server using a hardware security module (HSM) and establishing a secure session, using mutual authentication by symmetric keys and deriving session keys from the authentication.

“ ... This works very well, is secure but still lacks a secure way to connect the user to the secure element and the HSM. “

The user needs a way to provide their own secret, so that their identity can be validated. Thus we need a Personal Identification Number (PIN) entry application. We also need a method to display the transaction so that the user can make an informed decision about the contents and the metadata of the transaction.

This approach has merit because it is scalable, secure, but it also has some drawbacks. Almost no phones have an accessible secure element, either the phone has none, or it is not meant for general use. As such, you still need a secure pin entry and display application, with all the security and static cryptography involved. You also need to enroll the PIN entry and display application.

Core security assumption:

- *The link between pin entry, the transaction display application and the secure element is trusted.*

The app-HSM hybrid

A revolutionary approach: One of the biggest pain points is the loss of the trust anchor in the form of a secure element, and the lack of secure elements in acceptable form on the major (mobile) platforms. In seeking to restore this trust anchor, the app-HSM hybrid was born.



We claim that the app-HSM hybrid is at least just as good as the current best practice using a secure pin entry and display application as described above. But let us first start with the idea.

The core idea was the result of a problem in a completely different domain. While developing a secure access control system, one of the problems was the following; how can a 'Access-as-a-Service' Provider guarantee the correctness of the access rules database?

To solve this problem, so a 'Access-as-a-Service' Provider can give strong guarantees about the correctness of the contents of the access rule database, the following idea was developed: The owner of an access application using the services of such Access Service Providers has to approve each change of an access rule to this access rule database by going through the following steps:

1. Preparing the change request
2. Sending the change request to the mobile phone of the owner of this access application;
3. Showing the change to the owner of this access application on his pre-registered mobile phone/app;
4. Asking for approval by entering his PIN;
5. Completing the change after validating it and owner's PIN on the server.

We will now take you on a journey. Step by step we will show you how the solution was assembled from the best available approaches with the app-HSM hybrid solution as the result. We will additionally summarize the core assumptions inherent to the system and describe the remaining risks.

Ubiqu realized that, instead of an access rule database in the above example, this could be the private key of a user in a secure element on a mobile phone. Instead of a change to the access rules, it could be an arbitrary action, a transaction or a hash. So, instead of giving permission to make changes in the database, the concept was generalized so that the user provides permission to use his private key to perform any abstract action possible with private keys.

The realization then came that the above solution can be combined with the central server HSM solution and a secure entry and display application, as previously described under current best practices. Such a new approach actually removes the threat of malware taking control of the secure element, because a user now can always verify the content of a transaction.

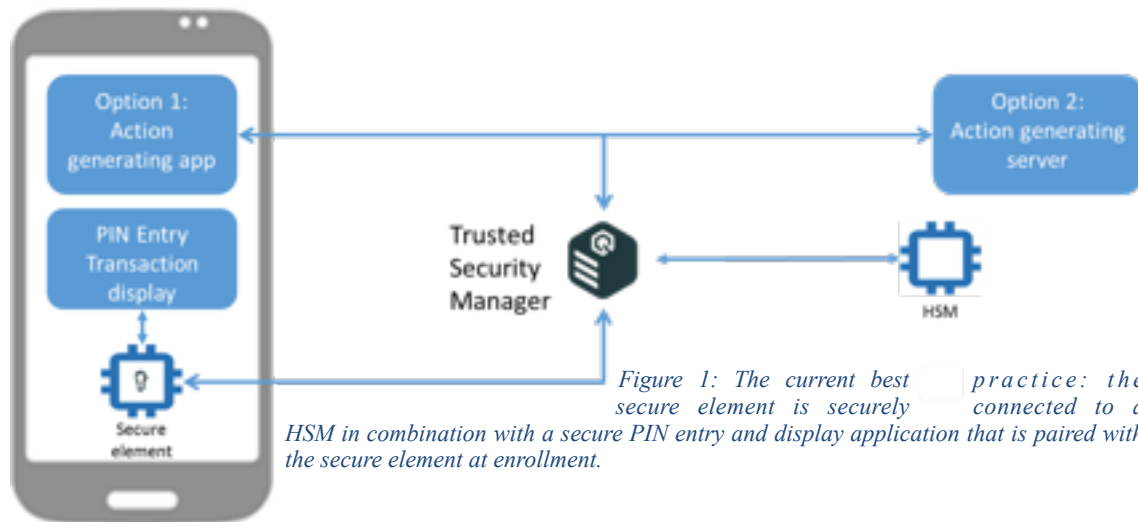


Figure 1: The current best practice: the secure element is securely connected to a HSM in combination with a secure PIN entry and display application that is paired with the secure element at enrollment.

“ One problem remains; when malware is present on the mobile device, the malware can initiate a transaction and can then approve this transaction, making the use of a secure element mute. ”

To prevent malware to be able to enter a pin and approve a transaction, even when it is capable of initiating a transaction, we added the following two ingredients:

1. Dynamic key update.
2. Secure Software, employing obfuscation, integrity checks, etc.

Now we have a solution that employs a secure element and that uses OOB tokens via the hardware security module (HSM) and the Trusted security manager (TSM).

The assumption is now:

- *The display and entry app cannot be hacked fast enough before a dynamic key update.*

The last statement is achieved because no static keys remain in the app. A hacker needs to hack the app to extract the dynamics keys used to secure the connection to the secure element.

The drawback is still:

- A secure element in the mobile phone is necessary;
- A malware app or hacker can hack the keyboard and sniff the PIN.



So we changed the order of PIN entry and transaction signing, where the transaction is first entered and shown and the PIN is tied to the transaction. In this way a real user always has to be involved, and malware injecting fake data is spotted early on. And added a customer keyboard.

The drawback is still:

- *A secure element in the mobile phone is necessary.*

So that is why we take the secure element and place it next to the HSM at TSM in the data center. This can be done without a problem because we have a secure channel between the app and its secure element.

To summarize we have:

- 1. A Secure element;**
- 2. Out of band tokens;**
- 3. Secure Software, employing obfuscation, integrity checks etc.;**
- 4. Dynamic key update.**

This leaves one remaining drawback:

The control app can be cloned by copying the app to another phone and patching a system call concerning device identifiable information.

**“...To prevent, or at least detect, cloning we added several countermeasures:
“**

- Device binding in the form of addressing: The server only sends messages to the device that is registered. Valid and usable addresses are mobile numbers and push tokens.
- Every software layer that is used adds and engages device identifiable information into the communication that is verified by the servers, TSM or HSM.



Core security assumption:

- The link between pin entry and transaction display application is trusted. Trust is achieved by making sure the app cannot be hacked fast enough before a key and app update occurs, which means the security keys have to be updated on a regular basis.
- Trust the addressing: trust that the app cannot be cloned and the messages cannot be intercepted without the user and/or the server noticing.
- We can update the app and the keys contained in it quickly and thoroughly enough, so that a hacker always has to start all over again after such an update.

This leaves one remaining drawback:

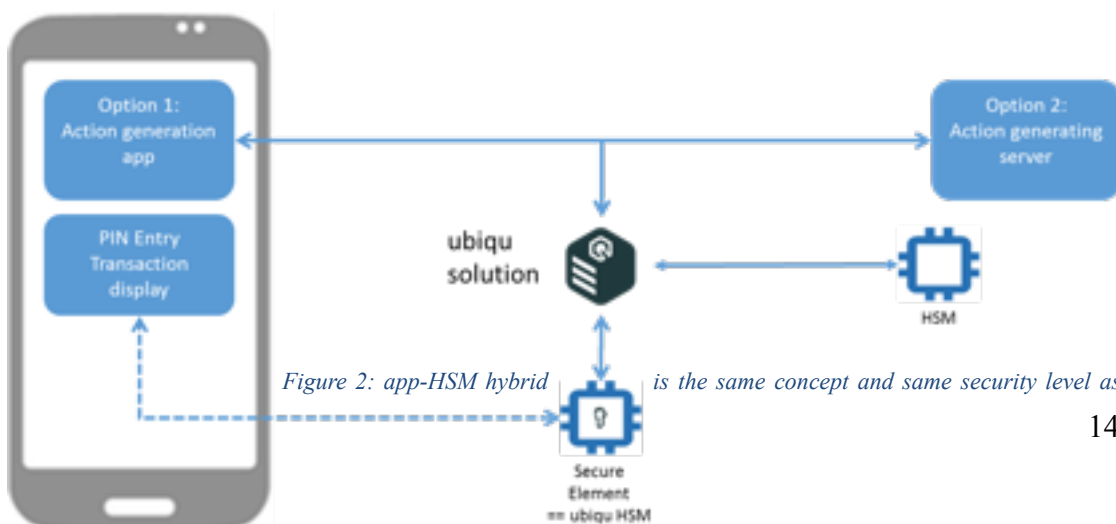
- If a hacker is able to write malware that in theory can hack an app fast enough, this malware will work for all apps on all phones.

To prevent, or at least detect cloning we added several countermeasures:

On enrollment, when generating the dynamic key and the code to protect the dynamic key, each app is made unique. Object code is made unique and all protection is individualized.

Core security assumption:

- Every app has some unique object code;
- A hacker cannot automate the generation of malware to reverse the custom obfuscation of each app.





best-in-class but also more flexible, cost effective and has the best user experience with high security on world population scale.

New benefits:

- We can now add a trust anchor to any device;
- On tamper detection, access to private key can be revoked;
- No offline hacking possibilities;
- Non-Repudiation.

“... We can now add a trust anchor to any device”

First, enrolling a device means first generating the first dynamic key and associating that dynamic key to a static key in the secure element in the data-center.

Second, associating that device to a user by use of an OOB registration code, allowing users to subsequently set their PIN.

Third, on tamper detection, access to private key can be revoked.

Last, all hackers are assumed to have a very high level of sophistication, access to many resources and a lot of time.

When a hacker wants to hack the solution they're left with four approaches:

1. Clone the app;
2. Hack the user interface and host OS;
3. Hack the app to retrieve all security keys and spoof other data.
4. Hack the server or even better hack the HSM

1. Clone the app

To clone the app the attacker needs to root the phone and copy all data to another phone. The attacker also has to find a way to receive the messages from the server. This means either having access to the SIM card or by having access to the push token of the device on a new, un-registered, device.



2. Hack the user interface or host OS

Malware could, as in the secure element scenario, try to approve a transaction on the device. In order to approve a transaction, it has to be able to start the app and enter the PIN. Since a custom keyboard is used, custom malware has to be crafted in order to hack the app.

Malware could intervene in the OS, on either the call level by manipulating data, spoofing identifiers and control flow, or by inputting data via text fields or other input methods. However, since data is decrypted in the app, the malware can control anything but the app itself, which reduces the viability of this approach even further.

3. Hack the app

Malware could copy the app. However, this should prove futile considering both the respectively weekly and monthly update cycle of the keys and key store, as well as the need to spoof the phones identifiers and grab the push / SMS messages from the original app's / phone's memory.

4. Hack the server

The HSM is certified to be protected in several ways against both physical and electronic tampering; as such it is impossible to extract the security keys from the HSM. Additionally, no transaction can be offered to the HSM without it first making a round trip to the app on the mobile phone and the user entering his PIN, so any hacker is also forced to get into the app, just controlling the server will not be enough.

In addition the following other approaches are futile:

- Side channel attacks on the app: A verified side channel resistant cryptographic implementation of the app is used. Side channel attacks on the server cannot be done without the user approving the transactions. The rate of successful transactions and the lack of control over faults results in the near impossibility of performing any side channel attack on the server.
- Change transactions on user device: User sees wrong transaction details. The malicious attacker would require control over the app, the app's OS, the customer's servers and the initiating app and/or browser of the user.
- Overlay user interface: Subsequent PIN entry by other app is blocked.
- Use a fake or cloned app: Once the app is enrolled, subsequent PIN entry by other code is blocked.
- Use a fake or cloned app on enrollment: Design the enrollment process in such a way that a user and you can verify that the user has installed the correct app before issuing /and or entering the activation code.

Could the Ubiqu app contain bugs or error resulting in malware?



Yes, ubiquitous software isn't bug free that would be an utopia, but exploitable bugs and errors are not present because:

1. The app code is individualized, so bugs are also in object code and app, but they manifest individually and can't be generalized to the whole population of apps.
2. The HSM codebase is small and scrutinized and of course certified, resulting in negligible chance of an exploitable error.



Conclusion

We have reintroduced a secure element in a revolutionary way.

“We can now employ a secure element on any device and provide a trust anchor for that device so that other programs can rely on that.”

Security is reduced to a hacker trying to hack an app, all other approaches are considered hard and out of reach of even the most sophisticated hackers.

The last remaining approach, i.e. hack the app, is made impossibly hard and also considered out of reach because: cloning doesn't work due to out of band addressing. Reverse engineering doesn't work because each app changes faster than the effort and thus time required to reverse engineer it. This is hindered further by the fact the



reverse engineering is not easily automated or parallelized; meaning more people trying doesn't necessarily speed up the process.

Lastly, each app is individual, meaning that a successful hack only impacts one user and one app.

- ubiqu -

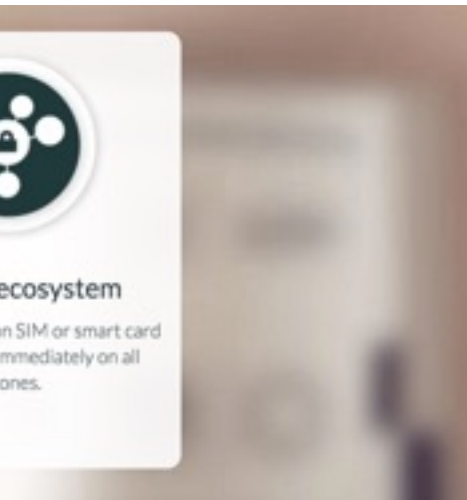


Author

Boris Goranov

Co-authors

- Tristan Timmermans
- Erwin Haasnoot
- Rogier Havermans
- Guus Stigter



“ Revolutionized the world of digital security “

General contact information

