SHAKE WELL BEFORE USE! Authentication based on Accelerometer Data

Rene Mayrhofer and Hans Gellersen, {rene, hwg}@comp.lancs.ac.uk

LANCASTER UNIVERSITY

ABSTRACT

Secure device pairing is especially difficult for spontaneous interaction in ubiquitous computing environments because of wireless communication, lack of powerful user interfaces, and scalability issues. We demonstrate a method to address this problem for small, mobile devices that does not require explicit user interfaces like displays or key pads. By shaking devices together in one hand for a few seconds, they are securely paired. Device authentication happens implicitly as part of the pairing process without the need for explicit user interaction "just for security". Our method has been implemented in two variants: first, for high-quality data collection using wired accelerometers; second, using built-in accelerometers in standard Nokia 5500 mobile phones.

THE PROBLEM

Device pairing over wireless channels is insecure because of the possibility of man-in-the-middle and impersonation attacks. To safeguard against such threats, device-to-device communication needs to be authenticated. However, in the case of spontaneous interaction, only the user who initiates the interaction can distinguish between the intended target and other similar devices or malicious attackers. Such an authentication is difficult because of two reasons: many devices lack explicit user interfaces like displays and keypads that could be used to verify the device pairing, and explicit authentication does not scale when considering hundreds of spontaneous interactions a day.

SHAKE WELL BEFORE USE!

We contribute a method for device-to-device authentication that is based on shared movement patterns which a user can simply generate by shaking devices together. Using embedded accelerometers, devices can recognize correlation of their movement and use movement patterns for authentication. From a user perspective, jointly shaking is a simple technique for associating devices. In our method, it simultaneously serves as out-of-band mechanism.

Users do not have to follow a particular pattern of shaking but that they can shake as they like; we do not attempt to identify people by their shaking patterns, but use it as a source of shared device movement.

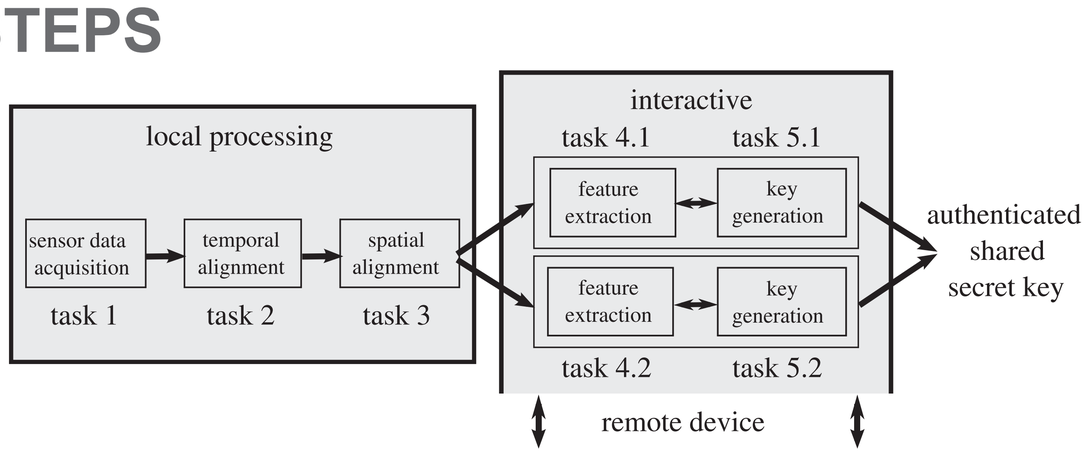
WHY SHAKING?

It is intuitive. People are familiar with shaking objects, for instance from shaking of medicine, or musical instruments – it can be done with minor attention.

It is vigorous. While there are many motion patterns that could be performed with two devices, shaking tends to produce the highest continuous acceleration values for as long as necessary to pair devices.

It is varying. The activity of shaking can be surprisingly different for different people, and thus generate high entropy from an attacker's point of view.

STEPS



Task 1: Sensor data acquisition
Sensor data is assumed to be available in the form of time series of acceleration values in all three dimensions, sampled at equidistant time steps. These must be taken locally and not be communicated wirelessly.

Task 2: Temporal alignment
As the two devices sample accelerometer time series independently in task 1, we require temporal synchronization for comparison. Triggering can be **explicit** or **implicit,** and synchronization either at **sample** or **event level**. Currently, we detect motion with high variance and align those parts of the time series where shaking is detected, which we call active segments, by their start times.

Task 3: Spatial alignment
Shaking is inherently a three-dimensional movement. In addition to the need to capture all three dimensions, the alignment between the two devices is unknown. This means that the three dimensions recorded by the two devices will not be aligned. We reduce the three dimensions to a single: by taking only the magnitude over all normalized dimensions, i.e. the length of the vector, we mitigate the alignment problem.

Task 4: Feature extraction
The problem of verifying that two devices are shaken, or more generally, moved together therefore becomes a classification problem. We currently use two related techniques for comparing time series from different devices: **coherence** between signals of 3 seconds length and multiple candidates of **pairwise added, exponentially quantized FFT coefficients** over sliding windows.

Task 5: Protocols
We have proposed two different protocols. Both have the same aim of generating an authenticated, secret shared key from sensor time series, but achieve this with very different designs.
The **first protocol** uses a conservative design and well-understood cryptographic primitives in two phases, key agreement and key verification. In the first phase, based on unauthenticated Diffie-Hellman key agreement, the devices agree to secret shared keys over the wireless channel. Using these keys in the second phase, they run an extended variant of the interlock protocol to exchange their recorded active segments in a way that detects man-in-the-middle attacks. Finally, the devices can locally and independently compare their local with the respective remote sensor data and verify if the pairing process should succeed or not. Our current implementation uses the coherence metric, but different devices can use different means of comparing active segments.
The second protocol is more unconventional and generates the secret shared key directly from sensor time series. The pairwise added, exponentially quantized FFT feature vectors are used as input to a Candidate Key Protocol, which broadcasts one-way hashes of the vectors as so-called candidate key parts. If a receiving device can compute the same one-way hash, it has verified that its sensor input matches that of the sender without actually revealing it to eavesdroppers. After a sufficient number of such matching key parts have been collected, they are concatenated and hashed again to create a so-called candidate key, which is again broadcast. If one or multiple remote devices can generate the same key, it is acknowledged and can be used for subsequent secure communication.
The **second protocol** is more dynamic and scalable, as it allows remote devices to "tune into" the key stream of another. On the other hand, the first protocol is more flexible in terms of using different methods of comparison and is considered more secure against offline attacks.